The TaskTracer System

Simone Stumpf, Xinlong Bao, Anton Dragunov, Thomas G. Dietterich, Jon Herlocker, Kevin Johnsrude, Lida Li, JianQiang Shen

School of Electrical Engineering
Oregon State University
Corvallis, OR
stumpf@eecs.oregonstate.edu

Abstract

Knowledge workers spend the majority of their working hours processing and manipulating information. These users face continual costs as they switch between tasks to retrieve and create information. The TaskTracer project at Oregon State University investigates the possibilities of a desktop software system that will record in detail how knowledge workers complete tasks, and intelligently leverage that information to increase efficiency and productivity. Our approach assigns each observed user interface action to a task for which it is likely being performed. In this demonstration we show how we have applied machine learning in this environment.

The TaskTracer System

Knowledge workers organize their work into discrete and describable units, such as projects, tasks or to-do items. The TaskTracer project at Oregon State University has developed a desktop system which operates in the Microsoft Windows environment, tracking most interactions with desktop applications as well as tracking phone calls. Once we have the past actions structured by task, we can provide substantial value to the knowledge worker in assisting in their daily task routines. Our goal is to develop five capabilities: more task-aware user interfaces in the applications we use daily, more efficient task-interruption recovery, better personal information management, workgroup information management and within-workgroup workflow detection and analysis. Our approach combines human-computer interaction and machine learning to assign each observed action (opening a file, saving a file, sending an email, cutting and pasting information, etc.) to a task for which it is likely being performed. In this demo, we show the current implementation of the TaskTracer system and its machine learning components, TaskPredictor and FolderPredictor.

There have been previous efforts to build environments that enable knowledge workers to manage multiple concurrent activities, which we call tasks, and use knowledge of those activities to improve productivity (Henderson and Card 1986, Freeman and Gelernter 1996, Dourish et al. 1999, Robertson et al. 2000, Bellotti et al. 2003). To be of assistance to a user, an agent (whether it is a computer system or a human associate) must "know" what the user is currently doing. In addition to the resources used in a task, a record of user's actions performed on those resources must also be captured to have the correct comprehension of the task context. There are systems (Fenstermacher and Ginsburg 2002, Kaptelinin 2003, Canny 2004) address this issue by aiming at recording as much information as possible about users' activities when they interact with computers. These activity records are obtained via monitoring the computer file system, input devices, and applications.

Our software, TaskTracer, employs an extensive data-collection framework to obtain detailed observations of user interactions in the common productivity applications used in knowledge work (see Dragunov et al. 2004). Currently, events are collected from Microsoft Office 2003, Microsoft Visual .NET, the Windows XP operating system and phone calls and stored as EventMessages in a database. In this framework, TaskTracer collects file pathnames for file create, change, open, print and save, text selection, copy-paste, windows focus, web navigation, phone call, clipboard and email events. Phone call data collection uses Caller Id to collect names and phone numbers of callers. In addition, speech-to-text software collects the user's — but not the caller's — phone speech.

Instead of using unsupervised clustering to discover tasks, users of TaskTracer manually specify what tasks they are doing in the initial stage of data collection, so that each action of the user (UI event) will be tagged with a particular task identifier to train predictors.

There are three main challenges to the machine learning approach. Firstly, accuracy must be exceptionally high to be acceptable to the user. Secondly, manual task switches have introduce "noise" into the task-tagged event stream. Thirdly, users may achieve the same task in different ways, hence doing something on the same task can generate different event streams. Conversely, different tasks may utilize the same objects, i.e. events and resources.

TaskPredictor

TaskPredictor is a component in TaskTracer that predicts the currently active task and sets the current task to the predicted task on the user's behalf. In contrast to plan recognition, our approach does not rely on an explicit induction of a plan or next events based on prior, sequential events. On-line learning is utilized to update the model if the user corrects the predicted task. The probabilistic framework we are employing in TaskPredictor can be outlined as follows. Suppose observation O = (o_{t-k} ,…, o_{t-1} , o_t) be an ordered set of observations from time t – k to t, where k is 0 if we ignore the temporal relationship and only consider the current observation. Our goal is to get a probabilistic distribution about the current task given O, P($Task_t$ = $task_i$ | o_{t-k} ,…, o_{t-1} , o_t). Feature construction occurs as follows. A Window-document segment (WDS) consists of the time period in which a window has the focus and this window is looking at a single document. It is assumed that the user is on a single task in the same WDS and a prediction is only necessary when the WDS changes. We make a prediction when navigation events occur in Internet Explorer, window focus switches, when a new application is started, or a resource is opened or saved. The source for the features comes from window titles, file pathnames, website URLs, and document content. Each source is segmented into a set of "words", where each word corresponds to a binary variable w_i in the feature vector. We utilize a stopword list to eliminate irrelevant features. We then use a Naïve Bayesian classifier to learn P(w | $task_i$) and P($task_i$) and make predictions by using Bayes rules to calculate P($task_i$ | w).

We have evaluated this approach by testing on a dataset collected from a team member, containing 81 different tasks, 11455 WDS and 1239 features. Always making a prediction, $\theta = 0$, where θ is the normalized probability of a task that is computed from the Naive Bayes model, provides an accuracy of 25%. Once "meaningless" events, such events that happen in all tasks (e.g. open/save dialogs, blank web pages) or events not related to any tasks (e.g. a file used by an application all the time) are removed, the accuracy can be further increased to 60%. Furthermore, we are investigating abstentions from making a prediction (i.e. imposing a threshold θ between 1 and 0), which increases the level of correct predictions to 85% ($\theta = 0.9$). It still remains to be investigated whether or not reduced coverage is good enough to the user, but not every WDS is a task switch so an abstention might be the right answer. We are currently exploring feature selection by using information gain which appears to push the accuracy to 95%.

FolderPredictor

Similarly, we have implemented a FolderPredictor to reduce the user cost of accessing resources given a certain task. Knowledge workers often have different folders for each task. In our approach we sort the folders based on the frequency in which user has opened/saved files before and use exponential decay to do a recency weighting. We currently employ these predictions by changing the navigation pane of the Windows Open or Save As dialog box. At the moment the usefulness of the prediction is evaluated by the cost to the user in reaching the desired file i.e. the distance between predicted folders and user's destination folder; it is on average one click away from the set of folders returned by FolderPredictor.

References

Bellotti, V., Ducheneaut, N., Howard, M. and Smith, I. 2003. Taking Email to Task: The Design and Evaluation of a Task Management Centered Email Tool. *Proceedings of the SIGCHI conference on Human factors in computing systems*, Ft. Lauderdale, Florida, ACM Press.

Canny, J. 2004. Gap: A Factor Model for Discrete Data. *Proceedings of SIGIR*, ACM Press.

Dourish, P., Edwards, K., LaMarca, A. and Salisbury, M. 1999. Presto: An Experimental Architecture for Fluid Interactive Document Spaces. *ACM Transactions on Computer-Human Interaction (TOCHI)* 6(2): 133-161.

Dragunov, A., Dietterich, T. G., Johnsrude, K., McLaughin, M., Li, L. and Herlocker, J. L. 2004. Tasktracer: A Desktop Environment to Support Multi-Tasking Knowledge Workers. *International Conference on Intelligent User Interfaces*, San Diego.

Fenstermacher, K. D. and Ginsburg, M. 2002. A Lightweight Framework for Cross-Application User Monitoring. *IEEE Computer* 35(3): 51-59.

Freeman, E. and Gelernter, D. 1996. Lifestreams: A Storage Model for Personal Data. *ACM SIGMOD Record* 25(1): 80-86.

Henderson, A. and Card, S. 1986. Rooms: The Use of Multiple Virtual Workspaces to Reduce Space Contention in a Window-Based Graphical User Interface. *ACM Transactions on Graphics (TOG)* 5(3): 211 - 243.

Kaptelinin, V. 2003. Umea: Translating Interaction Histories into Project Contexts. *Proceedings of the SIGCHI conference on Human factors in computing systems*, Ft. Lauderdale, Florida, ACM Press.

Robertson, G., van Dantzich, M., Robbins, D., Czerwinski, M., Hinckley, K., Risden, K., Thiel, D. and Gorokhovsky, V. 2000. The Task Gallery: A 3d Window Manager. *Proceedings of the SIGCHI conference on Human factors in computing systems*, The Hague, The Netherlands, ACM Press.