Detecting Real User Tasks by Training on Laboratory Contextual Attention Metadata

Andreas S. Rath [1]
arath@know-center.at

Didier Devaurs [1,2]
ddevaurs@know-center.at

Stefanie N. Lindstaedt [1,2]
slind@know-center.at

[1] Know-Center GmbH, Inffeldgasse 21a, 8010 Graz, Austria
[2] Knowledge Management Institute, Graz University of Technology, Graz, Austria

Abstract: Detecting the current task of a user is essential for providing her with contextualized and personalized support, and using Contextual Attention Metadata (CAM) can help doing so. Some recent approaches propose to perform automatic user task detection by means of task classifiers using such metadata. In this paper, we show that good results can be achieved by training such classifiers offline on CAM gathered in laboratory settings. We also isolate a combination of metadata features that present a significantly better discriminative power than classical ones.

1 Introduction

Today knowledge workers have to handle incessantly increasing amounts of information, in terms of text documents, emails, etc., located both on their own computer desktops and on the Web. Personal information management and search systems can help coping with this ever growing challenge, and they can do it even more efficiently if they provide contextualized and personalized support. For that they have to take account of the user's *usage and contextual attention metadata* [SLDH06, WNVD07, VS07, RWK+08, CDH+08] [1]. Such metadata is especially valuable for detecting the user's task, and several approaches have been proposed for doing so [OSTS06, SLDH06, LFBG07, GKS+08, RDL09]. By focusing on the user's interactions with applications and resources (*user interaction context*) we explore how to fully exploit this metadata to make user task detection more efficient in a real work environment.

Related Work: Here, by task detection we mean *task class detection* also referred to as task classification, as opposed to *task switch detection*. Task classification deals with the challenge of classifying usage data from user task execution into task classes or task types. Task switch detection involves predicting when the user switches from one task to another [OSTS06, SLDH06]. Automatic task detection is classically modeled as a machine learning problem, and more precisely a classification problem. This method is used to recognize Web based tasks [GCN08], tasks within emails [DLK06, SLDH06] or from the complete

[1] For us, CAM frameworks are not only approaches that use the CAM-XML data schema, but more generaly frameworks which observe contextual attention metadata independently of the underlying (sensor) data schema.

user's desktop [OSTS06, SLDH06, LFBG07, GKS+08, RDL09]. All these approaches are based on the following steps: (i) The CAM is captured by system and application sensors. (ii) Features, i.e. parts of this data, are chosen to build classification training instances. (iii) To obtain valid inputs for machine learning algorithms, these features are first transformed into attributes [WF05]. This transformation may include data preprocessing operations, such as removing stopwords [LFBG07, GKS+08] and application specific terms [OSTS06], or constructing word vectors. (iv) Attribute selection [SLDH06, GKS+08] is performed (optional step) to select the best discriminative attributes for (v) training the classification/learning algorithms.

Contribution: In this paper, while adopting this classical approach for task detection, we focus on investigating the following research questions: (i) How good can the performance of a task classifier be when used in a real work environment, while being trained with CAM gathered in laboratory settings? (ii) Which are the automatically observable CAM features that allow for good task detection performance? Both questions are concerned with work efficiency. The goal of the first one is to determine whether a task classifier can be trained offline, thus saving the user from the burden of manual training during work processes, which might slow down her computer and hence have a negative influence on her work efficiency and user experience. The second one aims at finding the most discriminative features among the automatically captured CAM features of our CAM framework for achieving a good balance between task detection accuracy and classification workload, which influences which CAM sensors have to be developed to do automatic task detection. To get a first impression on the answers to these research questions, we have designed a large-scale experiment in which users have performed a set of tasks both on a single laboratory computer and on their personal workstations. Our first results, achieved with a dataset containing four task classes and 203 task instances from 14 users, indicate that: (i) reliable detection of real tasks via offline training is possible, (ii) the good discriminative power of the classical *window title* feature [OSTS06, SLDH06, LFBG07, GKS+08] is confirmed, and (iii) classification accuracy is significantly increased by using a combination of six features specific to our approach.

Outline: In Section 2 we describe our experiment and our CAM framework. Our methodology for task classification is presented in Section 3, and the results obtained from the task classifier are discussed in Section 4. Finally we draw some conclusions in Section 5.

2 Experiment Design and Execution

Our experiment was carried out in the knowledge-intensive domain of the Know-Center. It was preceded by an analysis phase, during which several requirements were defined, by interviewing knowledge workers. Users required to know what kind of data was recorded, to be able to access and modify it, and that the evaluation results were anonymized. They could practice with the recording tool for a week before the experiment in order to reduce the bias of unfamiliarity during the experiment. This study/experiment was exploratory, the comparison was *within subjects* and the manipulations were achieved by the environment (laboratory vs. personal workstation) and the executed task (four different tasks).

Manipulation 1 - Work Environment: The first manipulation was achieved by varying the work environment (i.e. the computer desktop environment) of the participants. Each participant performed the same set of tasks both on a *laboratory computer* on which a set of standard software used in the company had been installed beforehand, and on their company *personal workstations* with their personal computer desktop settings and access to their personal files, folders, bookmarks, emails and so on. Half of the participants worked first on the laboratory computer and then on their personal workstations, and vice versa for the other half. The assignment of the participants to each group was randomized.

Manipulation 2 - Task: The second manipulation resulted from varying the tasks themselves. During a preliminary meeting, the participants of the experiment agreed on a selection of four tasks typical of the Know-Center domain: (1) "Filling in the official journey form", (2) "Filling in the cost recompense form for the official journey", (3) "Creating and handing in an application for leave" and (4) "Planning an official journey". [2] A short questionnaire was issued before starting the experiment to make sure that the probands understood the tasks they had to perform, and also to have them thinking about the tasks before they actually executed them.

During the experiment, for capturing the users' CAM, we have used multiple sensors [RWK+08] that record the user interactions with resources and applications on the computer desktop (see Table 1). The captured usage data is used to automatically populate our *User Interaction Context Ontology* (UICO) [RDL09] which constitutes an ontology-based user context model of the personal information management domain. Rule-based and information extraction techniques are used to automatically enrich our ontology, by discovering new instances of concepts, and deriving inter-concepts relationships [RDL09]. For a full presentation of the UICO, the interested reader is refered to [RDL09].

3 Task Classification Methodology

Our dataset contains 203 task instances from 14 users: 106 tasks from the laboratory computer and 97 from work station computers. The number of representatives for each task class were almost equally distributed (see Table 2). In our experiment we evaluated the influence of five parameters on the task detection performance: (i) the number of features, (ii) the classification model, (iii) the feature category, (iv) single features and (v) the combination of the k top performing single features with $k \in \{2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20\}$. For the task detection part we used the machine learning toolkit Weka [WF05] for parts of the data preprocessing, filtering, attribute selection, and classification.

Training/Test Instance Construction: Constructing training instances for the machine learning algorithms was done on the task level: each task represents a training instance for a specific class to be learned. A class corresponds to a specific task model. Having multiple task models hence resulted in a *multi-class* classification problem. An instance for a class was built from features and feature combinations based on the CAM captured

[2] It is worth noting that these tasks present different characteristics, in terms of complexity, estimated execution time, number of involved resources, granularity and so on.

Sensor	Observed Data and Metadata
Application Sensors	
Microsoft Word	document title, document url, language, text encoding, content of visible area, file name, folder, user name
Microsoft PowerPoint	document title, document url, document template name, current slide number, language, content, file name
Microsoft Excel	spreadsheet title, spreadsheet url, worksheet name, content of the currently viewed cell, authors, language, file name, file uri, folder, user name
Microsoft Internet Explorer	currently viewed url, urls of embedded frames, content as html, content as plain text
Microsoft Explorer	currently viewed folder/drive name, url of folder/drive path
Mozilla Firefox 2.x and 3.x	currently viewed url, urls of embedded frames, content as html
Mozilla Thunderbird	(html/plain text) content of currently viewed or sent email, subject, unique path (uri of email/news message) on server, user's mail action (compose, read, send, forward, reply), received/sent time, email addresses and full names of the email entries
Microsoft Outlook 2003/2007	(create, delete, modify, open and distribute) tasks, notes, calendar entries, contacts, data about email handling
Novell GroupWise email client	(create, delete, modify, and distribute) tasks, notes, calendar entries, todos, data about email handling
System Sensors	
File System Sensor	copying from/to, deleting, renaming from/to, moving from/to, modification of files and folders (file/folder url)
Clipboard Sensor	clipboard changes (i.e. text copied to clipboard)
Network Stream Sensor	header and payload content from network layer packets (http, ftp, nttp, smtp, messenger, ICQ, Skype...)
Generic Windows XP System Sensor	mouse movement, mouse click, keyboard input, window title, date and time of occurrence, window id/handle, process id, application name
Accessibility Object Sensor	name, value, description, help text, help text description, tooltip of the accessibility object.

Table 1: This table shows a listing of the developed application and system sensors as well as of the data recorded by these sensors. Application sensors are context sensors for standard office applications. System sensors are deep hooks into the operating system. They allow to record user interactions and network transfer data on a fine-granular level.

Set	Task 1	Task 2	Task 3	Task 4	Sum
Train	30	26	26	24	106
Test	25	19	25	28	97
Sum	55	45	51	52	203

Table 2: This table shows the distribution of training and test instances for the different task classes. Training and test instances were constructed based on the usage data recorded on the laboratory computer and on the personal workstations respectively.

during a task execution. For details about the data preprocessing and the feature to attribute transformation we refer to [RDL09].

We have defined 50 features that can be grouped in six categories: (i) ontology structure, (ii) content, (iii) application (iv) resource, (v) action and (vi) switching sequences. The *ontology structure category* contains features representing the number of instances of concepts and the number of datatype and objecttype relations used per task. The *content category* consists of the content of task-related resources, the content in focus and the text input of the user. The *application category* contains the classical *window title* feature [OSTS06, SLDH06, LFBG07, GKS$^+$08], the application name feature [GKS$^+$08] and our newly introduced graphical user interface elements (accessibility objects[3]) features. The *resource category* includes the complete contents and URIs (URLs) [SLDH06] of the used, referenced and included resources, as well as a feature that combines all the metadata about the used resources in a 'bag of words'. The *action category* represents the user interactions and contains features about the interactions with applications [GKS$^+$08], resources types, resources, key input types (navigational keys, letters, numbers), the number of events and event blocks, the duration of the event blocks, and the time intervals between event blocks. The *switching sequences category* comprises features about switches between applications, resources as well as event and resource types.

Classifiers and Attribute Selection: We studied the Naïve Bayes (NB), Linear Support Vector Machine (SVM) with cost parameter $c \in \{2^{-5}, 2^{-3}, 2^{-1}, 2^0, 2^1, 2^3, 2^5, 2^8, 2^{10}\}$[4], J48 decision tree (J48) and k-Nearest Neighbor (KNN-k) with $k \in \{1, 5, 10, 35\}$ algorithms. For each classifier/learning algorithm $l \in L$, for each feature category $\phi \in \Phi$ and each feature $f \in F$ we selected the g attributes having the highest *Information Gain* (IG) value to obtain our dataset. As values for g we used 50 different measure points. Half of them were equally distributed over the available number of attributes with an upper bound of 5000 attributes. The other half was defined by $G = \{3, 5, 10, 25, 50, 75, 100, 125, 150, 175, 200, 250, 300, 500, 750, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 5000, 7500, 10000\}$. Which attributes are finally used by the classifiers depends on the *attribute selection* algorithm (IG in our case). To answer our research questions we used the tasks we recorded from the laboratory computer as the training set and those recorded at the personal workstations as the test set. We measured the accuracy (a) of the used algorithms

[3]Microsoft Active Accessibility: http://msdn.microsoft.com/en-us/accessibility/

[4]The interval was chosen according to the libSVM guide at: http://www.csie.ntu.edu.tw/~cjlin/libsvm/

Set	R_S	f	l	g	a	p	r	R_G
	1	**Application Cat.**	**NB**	**500**	**91.75**	**0.97**	**0.92**	**3**
	2	Content Cat.	NB	500	86.60	0.95	0.87	5
Feature	3	All Categories	NB	750	82.47	0.93	0.83	8
categories	4	Resource Cat.	NB	1000	68.04	0.86	0.67	14
	5	Ontology Str. Cat.	J48	359	65.98	0.86	0.67	15
	6	Action Cat.	SVM	11	59.79	0.81	0.58	17
	7	Switching Seq. Cat.	NB	1832	40.21	0.66	0.39	20
	1	window title	J48	100	85.57	0.95	0.87	6
	2	content in focus	NB	150	84.54	0.94	0.84	7
	3	acc. obj. name	J48	100	80.41	0.92	0.81	9
Single	4	event block content	NB	200	73.20	0.89	0.76	10
features	5	acc. obj. value	J48	150	71.13	0.89	0.72	11
	6	UICO datatype relations	J48	221	70.10	0.88	0.71	12
	7	used resources metadata	J48	1000	68.04	0.86	0.68	13
	8	used resources content	NB	175	62.89	0.84	0.65	16
	9	content of resources	NB	250	58.76	0.81	0.61	18
	10	acc. obj. role	J48	31	54.64	0.79	0.55	19
Top k	**1**	**Top $k = 6$**	**NB**	**250**	**94.85**	**0.98**	**0.95**	**1**
single	**2**	**Top $k = 5$**	**NB**	**150**	**92.78**	**0.97**	**0.94**	**2**
features	3	Top $k = 4$	NB	500	89.69	0.96	0.91	4

Table 3: Overview of all results about the performance of detecting real workstation tasks by training on CAM from laboratory settings for each feature category, for all feature categories combined, each single feature as well as the k top performing single features. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), the ranking in the corresponding section (R_S) and across sections (R_G) are also given.

(l), the number of attributes (g), the micro precision (p) and micro recall (r).

4 Results and Discussion

An overview of all results about the performance of detecting real workstation tasks by training on CAM from laboratory settings is given in Table 3.

Feature Categories: The best feature category was the application category that correctly identified 91.75% of the real tasks (l=NB, g=500, p=0.97, r=0.92). Approximately 5% behind in terms of accuracy was the content category (l=NB, a=86.60%, g=500, p=0.95, r=0.87). Using all 50 features resulted in a 82.47% accuracy, which is about 9% worse than the best performing feature category. This directed our decision to study the used features in greater details.

Single Features: We evaluated the performance of each single feature separately and confirmed that the *window title* [OSTS06, SLDH06, LFBG07, GKS$^+$08] was the best dis-

criminative feature: it obtained an accuracy of 85.57% (l=J48, g=100, p=0.95, r=0.87). Of great interest are the good performances of accessibility object features: the *acc. obj. name* with a=80.41% (l=J48, g=100, p=0.92, r=0.81) and the *acc. obj. value* with a=71.13% (l=J48, g=150, p=0.89, r=0.72). Simply counting the number of UICO datatype relations (a=70.10%) was also quite efficient. Since the performance of the single features were that good we were wondering if we could do better by following the simple approach of combining the k best performing single features with respect to the accuracy (a).

Top *k* Features: We studied the performance for different k and obtained with the NB classifier at g=250 attributes with the *top 6* features the highest accuracy (a=94.85%), precision (p=0.98) and recall (r=0.95), among all studied features, feature categories and top k combinations. This was an accuracy increase of 9.28%, a precision increase of 0.03 and a recall increase of 0.08 compared to the performance of the window title feature. The number of attributes (250) of this best performing approach also supports prior work showing that a good choice for it is between 200 and 300 attributes [LFBG07, SLDH06].

Generalizability to Other CAM Frameworks: CAM frameworks that observe user contextual information differ in terms of utilized sensors and of granularity of the captured CAM. Our approach is very fine-granular since we observe not only the content currently viewed by the user or the window title of the application in focus but also the user's interactions with all desktop elements and application controls (accessibility objects). In our approach every single interaction of the user with an application and a resource is important and hence captured, stored and analyzed. Using a different CAM framework could result in leaving out "context features" with a good task discriminative power and could hence have a negative impact on the task detection performance.

Generalizability to Other Work Contexts: Our results only give a first impression on the fact that the environment in which users perform their tasks has no significant influence on the task detection performance. Since only four tasks were studied in our experiment, the generalizability of our results is of course limited, and further experiments (with other tasks, other users and in other domains) are needed. However it is well recognized that the *window title* feature has a good cross-domain discriminative power, and we think that it could be true for other CAM features.

Possible Impacts of our Results: The strong positive influence of specific context features on task detection performance could be an indication that it is not necessary to sense "everything" about the user's interactions with her computer desktop but only some relevant elements. This would have an impact on what kind of sensors have to be developed, i.e. which context features have to be sensed, to achieve a reasonable task detection performance. It would also impact the user's system performance because capturing less data normally leads to less cpu requirements. Furthermore if we know which features are performing well for supervised machine learning algorithms in laboratory settings, it could provide a first indication on which features could be used in an unsupervised learning approach and in real world settings. However, this would require further experiments in laboratory and real world settings.

5 Conclusions and Future Work

Following the CAM approach for observing the user's attention seems to be a good choice for automatically detecting the user's task by means of machine learning techniques. We have performed a large-scale experiment that shows that it is possible to obtain good task detection results for real user tasks with a classifier trained offline on laboratory CAM. We have also studied the discriminative power of individual and combined CAM features. The good performance of the classical *window title* feature was confirmed and even significantly outperformed by a specific combination of 6 features. Within this combination are CAM features that are specific to our ontology-based user interaction context model (UICO). This confirms that our model can improve task detection performance [RDL09].

We plan to study the discriminative power of our ontology-specific CAM features more thoroughly by performing further experiments with tasks having different characteristics, since the results we have obtained here are maybe domain-specific. We would like to understand why some features perform better than others for task detection. Our objective is to exhibit a small combination of CAM features with a strong discriminative power independently of the domain in order to enhance automatic task detection performance.

6 Acknowledgments

We would like to thank all the people from the Know-Center who participated in the experiment and supported us in carrying out our task experiment study.

The Know-Center is funded within the Austrian COMET Program - Competence Centers for Excellent Technologies - under the auspices of the Austrian Ministry of Transport, Innovation and Technology, the Austrian Ministry of Economics and Labor and by the State of Styria. COMET is managed by the Austrian Research Promotion Agency FFG.

References

[CDH+08] S. Chernov, G. Demartini, E. Herder, M. Kopycki, and W. Nejdl. Evaluating personal information management using an activity logs enriched desktop dataset. In *Workshop on Personal Information Management, CHI '08*, Florence, Italy, 2008.

[DLK06] M. Dredze, T. Lau, and N. Kushmerick. Automatically classifying emails into activities. In *IUI '06*, pages 70–77, Sydney, Australia, 2006.

[GCN08] A. Gutschmidt, C. H. Cap, and F. W. Nerdinger. Paving the path to automatic user task identification. In *Workshop on Common Sense Knowledge and Goal-Oriented Interfaces, IUI '08*, Canary Islands, Spain, 2008.

[GKS+08] M. Granitzer, M. Kröll, C. Seifert, A. S. Rath, N. Weber, O. Dietzel, and S. N. Lindstaedt. Analysis of machine learning techniques for context extraction. In *ICDIM '08*, pages 233–240, London, UK, 2008.

[LFBG07] R. Lokaiczyk, A. Faatz, A. Beckhaus, and M. Goertz. Enhancing just-in-time e-learning through machine learning on desktop context sensors. In *CONTEXT '07*, pages 330–341, Roskilde, Denmark, 2007.

[OSTS06] N. Oliver, G. Smith, C. Thakkar, and A. C. Surendran. SWISH: semantic analysis of window titles and switching history. In *IUI '06*, pages 194–201, Sydney, Australia, 2006.

[RDL09] A. S. Rath, D. Devaurs, and S. N. Lindstaedt. UICO: an ontology-based user interaction context model for automatic task detection on the computer desktop. In *Workshop on Context, Information and Ontologies, ESWC '09*, Heraklion, Greece, 2009.

[RWK+08] A. S. Rath, N. Weber, M. Kröll, M. Granitzer, O. Dietzel, and S. N. Lindstaedt. Context-aware knowledge services. In *Workshop on Personal Information Management, CHI '08*, Florence, Italy, 2008.

[SLDH06] J. Shen, L. Li, T. G. Dietterich, and J. L. Herlocker. A hybrid learning system for recognizing user tasks from desktop activities and email messages. In *IUI '06*, pages 86–92, Sydney, Australia, 2006.

[VS07] M. Van Kleek and H. E. Shrobe. A practical activity capture framework for personal, lifetime user modeling. In *UM '07 - Poster*, pages 298–302, Corfu, Greece, 2007.

[WF05] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.

[WNVD07] M. Wolpers, J. Najjar, K. Verbert, and E. Duval. Tracking actual usage: the attention metadata approach. *Educational Technology & Society*, 10(3):106–121, 2007.