# Exploiting the user interaction context for automatic task detection

Didier Devaurs, Andreas S. Rath, Stefanie N. Lindstaedt

# Exploiting the User Interaction Context for Automatic Task Detection

Didier Devaurs[1,2], Andreas S. Rath[3], and Stefanie N. Lindstaedt[3,4*]

[1] CNRS; LAAS; 7 avenue du colonel Roche, F-31077 Toulouse, France
[2] Université de Toulouse; UPS, INSA, INP, ISAE; LAAS; F-31077 Toulouse, France
[3] Know-Center GmbH[†], Inffeldgasse 21a, 8010 Graz, Austria
[4] Knowledge Management Institute, Graz University of Technology, Graz, Austria

**Abstract**

Detecting the task a user is performing on her computer desktop is important for providing her with contextualized and personalized support. Some recent approaches propose to perform automatic user task detection by means of classifiers using captured user context data. In this paper we improve on that by using an ontology-based user interaction context model that can be automatically populated by (i) capturing simple user interaction events on the computer desktop and (ii) applying rule-based and information extraction mechanisms. We present evaluation results from a large user study we have carried out in a knowledge-intensive business environment, showing that our ontology-based approach provides new contextual features yielding good task detection performance. We also argue that good results can be achieved by training task classifiers 'offline' on user context data gathered in laboratory settings. Finally, we isolate a combination of contextual features that present a significantly better discriminative power than classical ones.[1]

## 1 Introduction

Today knowledge workers have to handle incessantly increasing amounts of digital information, in terms of text documents, emails, multimedia files, etc., located on their own computer desktops, on company networks and on the World Wide Web. Personal information management, search and retrieval systems can help coping with this ever growing challenge. They can do so even more efficiently if they provide contextualized and personalized support. Various research areas have already emphasized the use of contextual information as one of the key elements for enhancing current applications. Examples can be cited in personal information management (Sauermann et al., 2005; Catarci et al., 2007; Chernov et al., 2008; Jones et al., 2008), user modeling (Van Kleek and Shrobe, 2007), information retrieval (Callan et al., 2007; Tang et al., 2007; Mylonas et al., 2008), technology-enhanced learning (Schmidt, 2005; Wolpers et al., 2007), etc.

An important issue in the context detection area is automatic user task detection on the computer desktop (Dey et al., 2001; Coutaz et al., 2005). If the user's current task is automatically detected, the user can be better supported with relevant information, such as learning and

work resources or task guidance. A classical approach has been to model task detection as a machine learning problem. However, the focus so far has been on using only text-based features and switching sequences (Oliver et al., 2006; Shen et al., 2007; Chernov et al., 2008; Granitzer et al., 2008) for detecting the user's task, which do not rely on ontology models. Furthermore controlled user studies and standard datasets for the evaluation of task detection approaches are still missing. This entails that the mechanisms underpinning the achievement of good task detection performance are yet to be unveiled.

In this paper we focus on (i) proposing new contextual features yielding improvements over the current results achieved by task detection techniques, and (ii) studying some aspects of the task detection problem in order to better understand in which settings it can be successfully applied. The first part of our contribution consists in proposing a generic ontology-based user context model for increasing the performance of user task detection. Our approach is based on using context sensors to capture simple interaction events (keyboard strokes and mouse clicks) from the user's computer desktop. Then, we utilize rule-based and information extraction techniques to automatically populate our user interaction context model by discovering instances of concepts and deriving inter-concepts relationships. Using an ontology-based user context model brings several advantages, such as an easy integration of new contextual attention metadata (Wolpers et al., 2007), a simple mapping of the raw sensor data into a unified model, and an easy extendability of the user context model with concepts and properties about new resources and user actions. We present an evaluation of our approach based on a large[2] controlled user study (containing five task classes and 220 task instances recorded from 14 participants) that we have carried out in a knowledge-intensive business environment. It shows that using an ontology-based representation of the user context allows to derive new ontology-specific features for machine learning algorithms, which increase their performance.

The second part of our contribution consists in investigating the following questions. (i) How good can the performance of a task classifier be when used in a real work environment, while being trained with contextual data gathered in laboratory settings? (ii) Which are the automatically observable contextual 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'. This would spare the user the burden of performing a manual training during work processes, which might slow down her computer and 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 contextual features, in order to achieve a good balance between task detection accuracy and classification workload. This would also influence which context sensors have to be developed to perform user task detection. To get a first impression on the answers to these questions, we have analyzed the classification results provided by the user study previously mentioned. In this study, users have performed their tasks both on a single laboratory computer and on their personal workstations. Our first results indicate that: (i) reliable detection of real tasks via 'offline' training is possible, (ii) the good discriminative power of the classical *window title* feature (Oliver et al., 2006; Shen et al., 2007; Granitzer et al., 2008) is confirmed, and (iii) classification accuracy is significantly increased by using a combination of six features specific to our approach.

The rest of the paper is organized as follows. First we describe how we define, conceptualize and model the user interaction context. We mainly focus on the presentation of our ontology-based context model. Then we elaborate on the sensors recording the interaction events and the techniques used to automatically populate the proposed ontology. Next we present the approach we follow to perform user task detection, and how we evaluate it. Our experimental results are discussed in the following section, including a comparison of several contextual feature sets in terms of task detection accuracy, as well as an analysis of several aspects of the task detection problem. Finally we provide concluding remarks and an outlook on future work.

[2]Involving 14 users, this study is much larger than previous studies reported in the field which involve only a few users (Oliver et al., 2006; Shen et al., 2007; Granitzer et al., 2008).

## 2 Modeling the User Interaction Context

Our view of the *"user context"* goes along with Dey's definition that context is *"any information that can be used to characterize the situation of entities that are considered relevant to the interaction between a user and an application, including the user and the application themselves"* (Dey et al., 2001). We specifically focus here on the *user interaction context* that we define as the interactions of the user with applications and resources on the computer desktop. With this definition in mind, we will sometimes simply use the term *user context* in the sequel. Our perspective puts the individual user and her actions into the center of attention. It aims at learning as much as possible about this individual and her actions. Our goal is to study the relations between the event objects generated by the user's interactions with the computer system, and their meaning and relevance to the user's task.

### 2.1 Conceptual Model - The Event Aggregation Pyramid

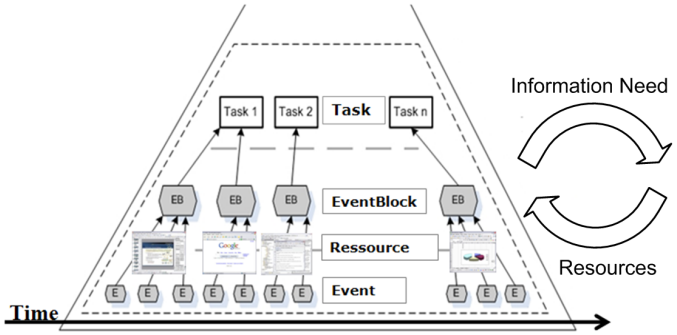


Figure 1: The event aggregation pyramid represents our conceptual view of the user interaction context.

The conceptual representation we propose for the user interaction context is the *event aggregation pyramid* (see Figure 1). Its layers represent the different aggregation levels of the user's actions. At the bottom of the pyramid are event objects (or simply events) that result from the user's interactions with the computer desktop. Above are aggregate events (or simply event-blocks), which are sequences of events that belong logically together, each event-block encompassing the user's actions associated with a specific resource acted upon. At the top are tasks, defined as groupings of event-blocks representing well-defined steps of a process. This model also integrates the idea of delivering resources that are relevant to the user's actions based on her information needs.

### 2.2 UICO - User Interaction Context Ontology

A context model is needed to store the user context data in a machine processable form. Various context model approaches have been proposed, such as key-value models, markup scheme models, graphical models, object oriented models, logic-based models, or ontology-based models (Strang and Linnhoff-Popien, 2004). The ontology-based approach has been advocated as being the most promising one mainly because of its dynamicity, expressiveness and extensibility (Strang and Linnhoff-Popien, 2004; Baldauf et al., 2007). In our specific case we argue that an ontology-based context model brings the following advantages: (i) It allows to easily integrate new context data sensed by context observers and to map the sensor data into a unified context model. (ii) It can be easily extended with concepts and properties about new resources and user actions. (iii) The relationships between resources on various granularity levels can be represented. (iv) The evolution of datatype properties (i.e. data and metadata) into objecttype

properties (i.e. relations between instances of ontology concepts) can be easily accomplished. (v) Being a formal model, it also allows other applications and services to build upon it and to access the encapsulated context information in a uniform way. Most importantly we show in the sequel that the performance of user task detection can be enhanced by using an ontology-based context model.

Our *User Interaction Context Ontology* (UICO) can be seen as the realization of the event aggregation pyramid with the support of semantic technologies. We follow a bottom-up approach and build the UICO by incrementally adding relations when new sensor data or algorithms are added. The UICO holds the contextual information representing the user interaction context. This includes the data provided by the context sensors observing the user's actions on the computer desktop and the information automatically derived from it. Based on the application domain in which the UICO is used we can decide which relations and concepts are useful or not. In the case of ontology-based task detection, we study concepts and relations that are significant for a specific task, i.e. highly discriminative between tasks. At the moment the UICO contains 88 concepts and 272 properties, and is modeled in OWL-DL[3] using the Protégé ontology modeling tool[4] (see Figure 2). From these 272 properties, 215 are datatype properties and 57 are objecttype properties. From a top level perspective, we define five different dimensions in the UICO: the *action dimension*, the *resource dimension*, the *information need dimension*, the *user dimension* and the *application dimension*.

**Action Dimension** The action dimension consists of concepts representing user actions, task states and connection points to top-down modeling approaches. User actions are distinguished based on their granularity, corresponding to the levels of the event aggregation pyramid: `Event` at the lowest level, then `EventBlock` and then `Task`. The `ActionType` concepts specify which types of actions are defined on each level. Currently we only distinguish action types on the event level (`EventType` concept): there are 25 of them (`Open`, `Save`, `Print`, `Copy`, `Reply`, `ClipboardChange`, `DesktopSearch`, etc.). As an example, if the user clicks on the search button of a search engine's web page, this user interaction will generate an `Event` of type `WebSearch`. The `TaskState` concept and its sub-concepts model the way the user manages and executes tasks. We borrow the types of task states from the *Nepomuk Task Management Model* (Groza et al., 2007). These types allow to model a user creating, executing, interrupting, finishing, aborting, approving and archiving a task. Besides, each change in task state is tracked via the `TaskStateChange` concept. The `Model` concept has been introduced to have connection points to top-down modeling approaches. Currently only one connection point is available: the `TaskModel` concept. This concept is similar to what is defined in the areas of workflow management systems and task process analysis. At the moment, the `TaskModel` concept is used to categorize a task. An example of instances of the `TaskModel` and `Task` concepts is "Planning a journey" and "Planning the journey to CHI 2011" respectively.

**Resource Dimension** The resource dimension contains concepts for representing resources on the computer desktop. We specifically focus on modeling resources commonly used by the knowledge workers we have interviewed. We have identified 16 key resource concepts (email, person, image, file, folder, etc.) but more can be easily added if required. A resource is constructed from the data and metadata captured by the context sensors. Relations can be defined between concepts of the resource dimension and of the action dimension for modeling on which resources what kind of user actions have been executed. For example, if the user enters a text in a Microsoft Word document, all keyboard entries will be instances of the `Event` concept, connected via the `isActionOn` objecttype property to the same instance of a `TextDocument` (and a `FileResource`) representing that document.

[3] http://www.w3.org/2004/OWL
[4] http://protege.stanford.edu

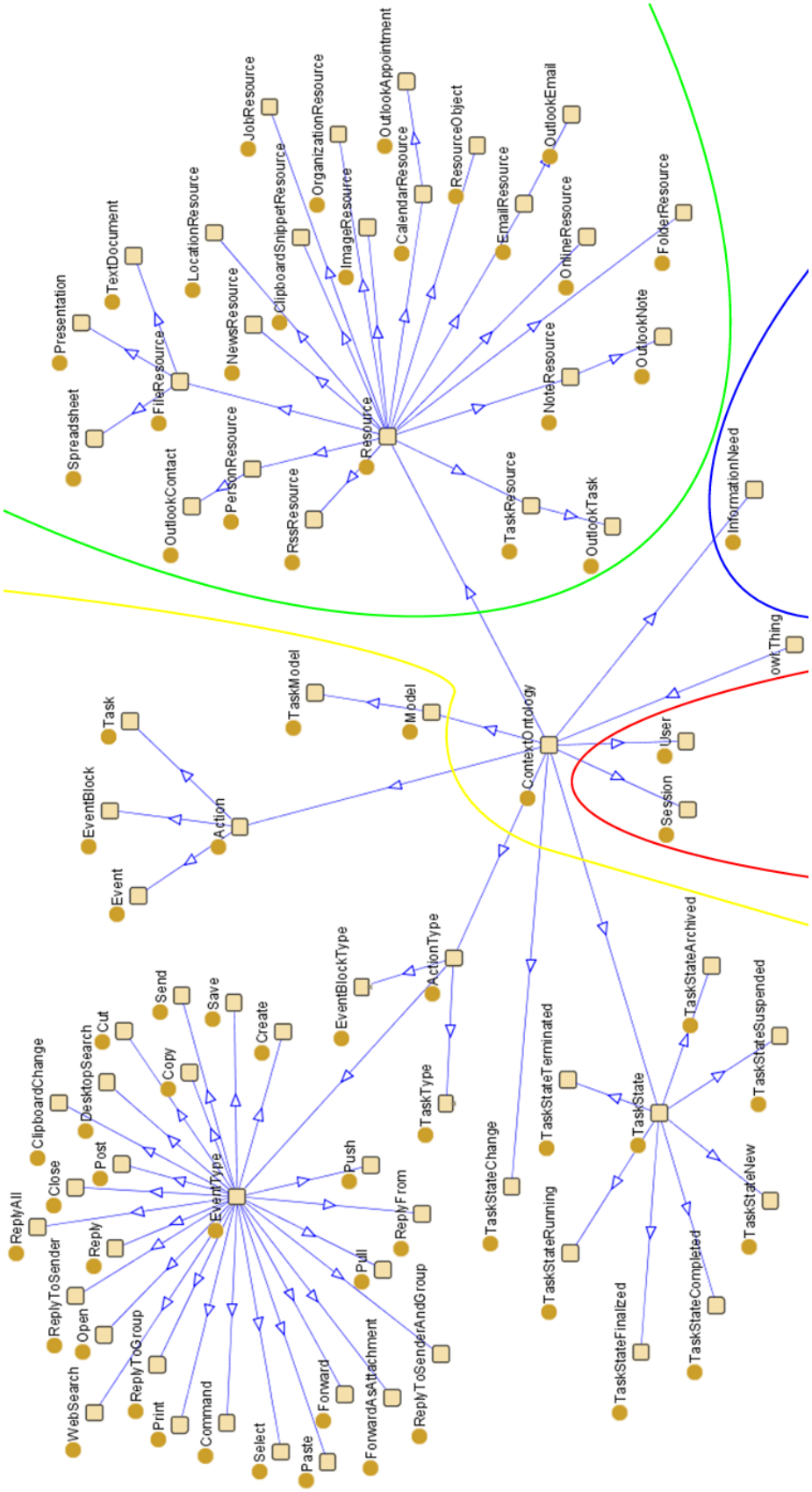


Figure 2: The concept hierarchy of the User Interaction Context Ontology (UICO) visualized with the Protégé tool. The left area contains the *action dimension*, the right area contains the *resource dimension*, and the bottom area contains the *user dimension* and the *information need dimension*.

**Information Need Dimension** The information need dimension represents the context-aware pro-active information delivery aspect of the UICO. An information need is detected by a set of fixed rules, based on the available contextual data. An `InformationNeed` concept has properties defining the accuracy of the detection and the importance to fulfill this need within a certain time-frame. An information need is associated with the user's action(s) that trigger(s) it, thus creating a connection between the information need dimension and the action dimension. The information need dimension is also connected to the resource dimension, since each resource that helps in fulfilling the user's information need is related via the objecttype property `suggestsResource` to the `InformationNeed`.

**User Dimension** The user dimension contains only two concepts: `User` and `Session`[5]. Its main interest is to maintain a link between a user and the interaction events she performs. The `User` concept defines basic user information such as user name, password, first name and last name. The `Session` concept is used for tracking the time of user logins and the duration of a user session in our application. The user dimension is connected to the action dimension in that each `Action` is associated with a `User` via the objecttype relation `hasUser`. It is also indirectly related to the resource and information need dimensions via the action dimension.

**Application Dimension** The application dimension is a "hidden" dimension, since it is not modeled as concepts in the UICO. It is however present in that each user interaction happens within the focus of a certain application, such as Microsoft Word or Windows Explorer. The `Event` concept holds the information about the user interaction with the application through the datatype properties `hasApplicationName` and `hasProcessId`. Standard applications that run on the Microsoft Windows desktop normally consist of graphical user interface (GUI) elements. Console applications also contain GUI elements such as the window itself, scroll bars and buttons for minimizing, maximizing and closing the application. Most of GUI elements possess an associated *accessibility object*[6] that can be accessed by context sensors. Datatype properties of the `Event` concept hold the data about the interactions with GUI elements. We show later on that these accessibility objects play an important role in task detection. A resource is normally accessed by the user within an application, hence there exists a relation between the resource dimension and the application dimension. This relation is indirectly captured by the relation between the resource dimension and the action dimension, i.e. by the datatype property `hasApplicationName` of an `Event`.

## 2.3 Related Ontologies

The UICO is similar to the *Personal Information Model Ontology* (PIMO) (Sauermann et al., 2007) developed in the NEPOMUK research project[7], in terms of representation of desktop resources. However, they differ in terms of granularity of concepts and relations. The UICO is a fine-grained ontology, driven by the goal of automatically representing low-level captured contextual information, whereas the PIMO enables the user to manually extend the ontology with new concepts and relations, to define her environment for personal information management. The *native operations* (NOP) ontology[8], used in the Mymory project (Biedert et al., 2008), models native operations (e.g., `AddBookmark` or `CopyFile`) on *generic information objects* (e.g., email, bookmark or file) recorded by system and application sensors. Native operations are similar to the UICO's `ActionType` concepts, and more specifically to the `EventType` concepts. The `DataObject` concepts describe several desktop resources in a more coarse-granular way

[5]The user dimension should not be mistaken with a user profile containing preferences, etc. Maintaining such a profile is out of the scope of this work.

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

[7]http://nepomuk.semanticdesktop.org

[8]http://usercontext.opendfki.de/wiki/NopOntology

than we do for the UICO's `Resource` concepts. In (Xiao and Cruz, 2005) a layered and semantic ontology-based framework for personal information management following the principles of *semantic data organization*, *flexible data manipulation* and *rich visualization* is proposed. The framework consists of an *application layer*, a *domain layer*, a *resource layer* and a *personal information space*. The resource dimension of the UICO can be seen as a combination of these domain and resource layers, since resource instances are mapped to concepts of the domain layer.

# 3 Automatic Population of the Context Ontology

It is not realistic to have a user manually providing the data about her context on such a fine-granular level, as is defined in our UICO. Hence semi-automatic and automatic mechanisms are required to ease the process of "populating" the ontology. We use rule-based, information extraction and machine learning techniques to automatically populate the ontology and automatically derive relations between the model's entities. We now describe how we build instances of concepts and augment relations between the concept instances. We also show which kind of sensors we use to observe user interaction events on the computer desktop, how we discover resources the user has utilized, unveil connections between resources, and aggregate single user interaction events into event-blocks and tasks.

## 3.1 Context Observation

Context observation mechanisms are used to capture the user's behavior while working on her computer desktop, i.e. performing tasks. Simple operating system and application events initiated by the user while interacting with her desktop are recorded by context observers, acting as event object sources. Our use of context observers is similar to the approach followed by contextual attention metadata (Wolpers et al., 2007) and other context observation approaches (Dragunov et al., 2005; Van Kleek and Shrobe, 2007). Context observers, also referred to as context sensors, are programs, macros or plug-ins that can be distinguished based on the origin of the data they deliver. *System sensors* are deep hook into the operating system. *Application sensors* are developed for standard office applications. We focus on supporting applications that knowledge workers use in their daily work, which is generally in a Microsoft Windows environment. Table 1 presents a list of the available system and application sensors, and a description of what kind of contextual information they are able to sense. The produced events are sent as an XML stream via an event channel to the context capturing framework (i.e. our event processing agent) for storage[9], processing and analysis. We also refer to this contextual attention metadata stream as the *event stream*. Our event processing network is rather a static one, evolving only when new sensors are added.

## 3.2 Resource Discovery

The resources that populate the ontology model are for example links, web pages, persons, organizations, locations, emails, files, folders, text documents, presentations or spreadsheets. Resource discovery is about the identification of resources and the extraction of related metadata in the event stream. It is also about unveiling the resources the user has interacted with, and the resources that are included or referenced in a used resource. We say that a resource is *included* in another one if its content is part of the content of another resource, e.g. when copying some text from an email to a text document. A resource is *referenced* by another one if its location appears in the content of another resource, e.g. when a link to a web page appears

[9]We store the recorded contextual data in a triple store, and more precisely a quad store featuring named graphs and SPARQL query possibilities.

| 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: List of our application and system sensors, and data recorded by these sensors.

in the content of an email. The resources identified by the resource discovery mechanisms are related to instances of the `Event` concept by the `isActionOn` objecttype property.

We apply three different techniques to discover resources: regular expressions, information extraction and direct resource identification. (i) The *regular expression* approach identifies resources in the event stream based on certain character sequences predefined as regular expressions. This is used to identify files, folders, web links or email addresses. (ii) The *information extraction* approach extracts person, location and organization entities in text-based elements in the event stream. This extraction is rule-based and utilizes natural language specifics. The extracted entities are mapped to concepts of the UICO, based on the available contextual information. As an example, when the name of a person is identified in a text document, it is mapped to an instance of a `Person` concept and a relation specifying that this person is mentioned in that document is built. (iii) The *direct resource identification* approach finds data about the resource to build directly in the sensor data, and directly maps certain fields of the event stream data to the resource. An example is the `ClipboardSnippetResource`, which is built from the content of the clipboard application sensed by the clipboard observer. Another example is the sensor data about an email opened by the user. In this case the sensor sends the information that a specific email identified by its `server message id` has been opened for reading. Additional metadata about the email is attached by the sensor and added to the discovered resource.

### 3.3 Event to Event-Block Aggregation

Context sensors observe low-level contextual attention metadata that result in simple events. For *logically* aggregating events that belong together (i.e. grouping user's actions) into blocks of events, so-called event-blocks, static rules are used. "Logically" here implies grouping the events that capture the interactions of a user with a single resource. Resources can be of various types and opened in different applications. Therefore, for different types of applications, different rules are applied in the grouping process. An application can handle multiple resource types, as is the case, e.g. for Microsoft Outlook or Novell GroupWise, in which emails, tasks, notes, appointments and contact details are managed. The complexity and accuracy of the static rules depend on the application mechanisms for identifying a single resource and on the possibility to capture this resource id with a sensor. If it is not possible for a sensor to capture a unique resource id in an application, heuristics are used to uniquely identify the resource.

Two types of rules can be distinguished for the event to event-block aggregating process. The first type is a set of rules designed for specific applications, and is referred to as *application specific rules*. An example of such a rule is: aggregate all events that happened on the same slide in the Microsoft PowerPoint application. The second type of rules, referred to as *default application rules*, are applied if no application specific rule is applicable. They also serve as backup rules, when there is not enough information in the event stream to apply an application specific rule. The goal of these rules is to heuristically aggregate events into event-blocks, based on event attributes that can be observed operating-system-wide by the context sensors. These attributes are the window title of the application, the process number and the window handle id[10]. The window title and the process number perform best for a generic event to event-block aggregation in which no application specific attribute is present. The discriminative power of the window title has been observed in other work as well (Oliver et al., 2006; Shen et al., 2007; Granitzer et al., 2008).

### 3.4 Event to Task Aggregation

Aggregating interaction events into tasks cannot be done with the previous rule-based approach, since it would require to manually design rules for all possible tasks. This might be a reasonable approach for well-structured tasks, such as administrative or routine tasks, but is obviously not appropriate for tasks involving a certain freedom and creativity in their execution, i.e. for knowledge-intensive tasks such as "Planning a journey" or "Writing a research paper". A solution is to automatically extract tasks from the information available in the user interaction context model by means of machine learning techniques. Once detected, these tasks will enrich the ontology model.

## 4 User Task Detection

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 executions into task classes or task types. Task switch detection involves predicting when the user switches from one task to another (Oliver et al., 2006; Shen et al., 2007).

### 4.1 Task Classification Approach

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 (Kellar and Watters, 2006; Gutschmidt et al., 2008), tasks within emails (Kushmerick and Lau, 2005; Dredze et al.,

[10]The window handle is a unique window identifier, constructed by the Microsoft Windows operating system.

2006; Shen et al., 2007) or from the complete user's desktop (Oliver et al., 2006; Shen et al., 2007; Granitzer et al., 2008). All these approaches are based on the following steps: (i) The contextual data is captured by system and application sensors. (ii) Features, i.e. parts of this data, are chosen to build classification training/test instances at the task level: each task represents a training/test instance for a specific class (i.e. a task model) to be learned. (iii) To obtain valid inputs for machine learning algorithms, these features are first subjected to some feature engineering (Witten and Frank, 2005), which may include data preprocessing operations, such as removing stopwords (Granitzer et al., 2008) and application specific terms (Oliver et al., 2006), or constructing word vectors. (iv) Feature-value selection (Witten and Frank, 2005; Shen et al., 2007; Granitzer et al., 2008) is (optionally) performed to select the best discriminative feature values. (v) Finally, the classification/learning algorithms are trained/tested on the training/test instances built from the feature values. Having multiple task models results in a *multi-class* classification problem. In this work, we also adopt this classical approach for task detection. We use the machine learning toolkit Weka (Witten and Frank, 2005) for parts of the feature engineering (steps (ii) to (iv)), and classification (step (v)).

### 4.2 Feature Engineering

Based on the UICO, we have defined 50 features that can be grouped in six categories: ontology structure, content, application, resource, action and 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 (Oliver et al., 2006; Shen et al., 2007; Granitzer et al., 2008), the application name feature (Granitzer et al., 2008) and our newly introduced GUI elements (accessibility objects) features. The *resource category* includes the complete contents and URIs (URLs) (Shen et al., 2007) 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 (Granitzer et al., 2008), resource 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 and event or resource types.

The process of transforming the event attributes associated with our 50 features into feature values that are usable by machine learning algorithms (step (iii) of the task classification approach) is referred to as *feature engineering* (Witten and Frank, 2005). The following steps are performed to preprocess the content of text-based features (in this sequence): (i) remove end of line characters, (ii) remove markups such as `\&lg` or `![CDATA`, (iii) remove all characters but letters, (iv) remove German and English stopwords, (v) remove words shorter than three characters. For each text-based feature, we generate values as vector of words with the `StringToWordVector` function of Weka. For example, for the *window title* feature, possible generated values are "{*review, Notepad*}", "{*Adobe, Reader, ontology*}", "{*firefox, google*}", "{*Microsoft, Word, Document*}", etc. For numeric features, we apply the Weka `PKIDiscretize` filter to generate values as intervals rather than numbers.

### 4.3 Evaluation Methodology

Evaluating a task detection approach is complex and requires assessing the impact of the various factors involved. In this work, we evaluate the influence of the following parameters on the task detection performance: (i) the set of used features, (ii) the number of used values generated from the features and (iii) the learning algorithm (classifier). The set of used features is varied by including (i) each feature individually, (ii) each feature category individually, (iii) all feature categories or (iv) the *top k* best performing single features, with $k \in \{2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20\}$.

To build a task instance, we select the $g$ feature values having the highest *Information Gain* (IG) (Witten and Frank, 2005), where $g$ is varied among 50 different integers distributed between 3 and 10000. Half of these integers are equally distributed between 3 and the number of available feature values, with an upper bound of 5000. The other half is 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$\}$, the upper bound being 10000 or the maximum number of available feature values, whichever is less. We have chosen to use IG to select feature values because it is one of the fastest and most popular methods in the text classification field. Comparing it with other feature selection methods is out of the scope of this paper. We use the IG computation functionality of Weka. For a given feature value $v$ and a given set of classes $C$, we have

$$IG(v) = \sum_{c \in C} P(v,c)\, log \frac{P(v,c)}{P(v)\, P(c)} \tag{1}$$

The classifiers we evaluate are Naïve Bayes (NB), J48 decision tree (J48), k-Nearest Neighbor (KNN-k) with the number of neighbors $k \in \{1, 5, 10, 35\}$, and Linear Support Vector Machine (SVM-c) with cost parameter $c \in \{2^{-5}, 2^{-3}, 2^{-1}, 2^0, 2^1, 2^3, 2^5, 2^8, 2^{10}\}$[11]. The Weka machine learning library (Witten and Frank, 2005) and the Weka integration of the libSVM[12] provide the necessary toolkit to evaluate these algorithms. For J48 decision tree, we use the default version of Weka, which performs some pruning (using the subtree raising approach) but no error pruning. To evaluate the learning algorithms, we perform a stratified 10-fold cross-validation (Witten and Frank, 2005). Besides, the training set and test set instances are strictly parted (i.e. constructed and preprocessed independently) to avoid any bias. Finally, we compute the mean values across all folds of the achieved accuracy, micro precision and micro recall.

# 5 Experimental Results

## 5.1 Experiment Design

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 unfamiliarity bias. This study was exploratory, the comparison was *within subjects* and the manipulations were achieved by the working environment (laboratory vs. personal workstation) and the executed task (five different tasks).

The first manipulation was achieved by varying the working 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.

The second manipulation resulted from varying the tasks themselves. During a preliminary meeting, the participants of the experiment agreed on a selection of five 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",

[11] These values are borrowed from the libSVM practical guide: http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[12] http://www.cs.iastate.edu/~yasser/wlsvm/

(4) "Planning an official journey" and (5) "Organizing a project meeting". 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. 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 think about the tasks before they actually executed them.

The dataset gathered during the experiment contains 220 task instances recorded from 14 users. Each user was supposed to perform all five tasks at least twice within both working environments, which should have produced at least 56 instances for each task class, and a total of at least 280 task instances. However, some instances were lost because of technical problems, and some were simply not performed by the users. The representatives of each task class are almost equally distributed (see Table 2) except for Task 5. Since most users expressed that this task was too difficult to perform on the laboratory computer (due to the lack of personal calendar, files, e-mails, etc.), we discard it in some parts of the subsequent analysis.

| | **Task 1** | **Task 2** | **Task 3** | **Task 4** | **Task 5** |
|---|---|---|---|---|---|
| *Laboratory computer* | 30 | 26 | 26 | 24 | 3 |
| *Personal workstations* | 25 | 19 | 25 | 28 | 14 |
| *Sum* | 55 | 45 | 51 | 52 | 17 |

Table 2: Distribution of the task instances with respect to the task class and the environment.

## 5.2 Comparison of Contextual Feature Sets

Since comparing different approaches cannot be done directly because of the difference in the granularity of training instances, we have decided to focus our comparison on the feature engineering part. We compare the performance of various combinations of contextual features, in terms of task detection accuracy. We use features that are specific to our ontology-based approach, and more classical features, as used in the TaskPredictor (Shen et al., 2007), SWISH (Oliver et al., 2006) and Dyonipos (Granitzer et al., 2008) systems. When using features from other approaches, we preprocess them based on the information available in the papers mentioned above, and evaluate the performance according to our experimental setup.

We consider all task instances recorded for Tasks 1 to 5, without taking into account the difference in working environment, which corresponds to the number of instances presented in the last row of Table 2. To evaluate the classification results, we apply a stratified 10-fold cross-validation, which varies the partition of the task instances between the training set and the test set. Table 3 presents a comparison of the best algorithm runs in terms of task detection accuracy for various features and feature categories. It shows that the best performing feature set is the *Top 4 features*, as defined in our UICO approach, namely the *accessibility object name*, the *window title*, the *accessibility object value* and the *content in focus*. The *window title* is a very classical feature used in all other approaches, but the other features are new and specific to our approach. The paired t-tests we have performed show that the *Top 4 features* performed statistically significantly better than all other features and feature combinations on a $p < 0.05$ significance level. The results in Table 3 also highlight that the J48 decision tree and the Naïve Bayes algorithms perform much better than the others. From the paired t-tests computed based on the classifiers achieved accuracy, we can derive the following partial order of these classifiers: $\{J48, NB\} \gg \{KNN\text{-}1, KNN\text{-}5, KNN\text{-}10, KNN\text{-}35\} \gg \{SVM\}$, with $\gg$ indicating a statistical significance on a $\alpha = 0.005$ level.

| | $f$ | $l$ | $g$ | $a$ | $p$ | $r$ |
|---|---|---|---|---|---|---|
| | All Categories | J48 | 750 | 86.71 | 0.96 | 0.85 |
| | Application Category | J48 | 50 | 86.71 | 0.96 | 0.84 |
| | Resource Category | NB | 3000 | 66.49 | 0.89 | 0.72 |
| | Content Category | NB | 4000 | 65.15 | 0.87 | 0.65 |
| | Ontology Structure Category | J48 | 359 | 63.38 | 0.86 | 0.64 |
| | accessibility object name | J48 | 50 | 86.21 | 0.96 | 0.86 |
| | window title | J48 | 75 | 81.26 | 0.94 | 0.81 |
| | accessibility object value | J48 | 25 | 71.15 | 0.90 | 0.69 |
| | content in focus | J48 | 1500 | 64.72 | 0.87 | 0.64 |
| | content of event-block | KNN-1 | 75 | 63.74 | 0.87 | 0.66 |
| UICO | **Top 4 features** | **J48** | **5000** | **88.55** | **0.97** | **0.87** |
| | Top 5 features | J48 | 1000 | 88.53 | 0.97 | 0.87 |
| | Top 2 features | KNN-10 | 25 | 88.12 | 0.96 | 0.89 |
| | Top 20 features | J48 | 500 | 88.07 | 0.96 | 0.87 |
| | Top 3 features | J48 | 3500 | 87.66 | 0.96 | 0.88 |
| | Top 7 features | J48 | 300 | 87.19 | 0.96 | 0.85 |
| | Top 6 features | J48 | 250 | 87.16 | 0.96 | 0.86 |
| | Top 8 features | J48 | 3000 | 87.14 | 0.96 | 0.84 |
| | Top 10 features | J48 | 300 | 86.73 | 0.96 | 0.85 |
| | Top 15 features | J48 | 500 | 86.28 | 0.96 | 0.85 |
| | Top 9 features | J48 | 300 | 85.76 | 0.96 | 0.83 |
| | $ApplicationName$ ($\mathcal{A}$) | J48 | 24 | 48.20 | 0.78 | 0.52 |
| | $Content$ ($\mathcal{C}$) | KNN-1 | 125 | 63.38 | 0.86 | 0.63 |
| | $WindowTitle$ ($\mathcal{W}$) | NB | 100 | 78.87 | 0.93 | 0.80 |
| Dyonipos | $\mathcal{A}\,\mathcal{C}$ | J48 | 150 | 68.27 | 0.89 | 0.69 |
| | $\mathcal{A}\,\mathcal{W}$ | J48 | 100 | 80.28 | 0.94 | 0.80 |
| | $\mathcal{C}\,\mathcal{W}$ | NB | 750 | 82.58 | 0.95 | 0.85 |
| | $\mathcal{A}\,\mathcal{C}\,\mathcal{W}$ | NB | 300 | 83.51 | 0.95 | 0.85 |
| SWISH | $\mathcal{W}$ | J48 | 150 | 79.35 | 0.93 | 0.78 |
| TaskPredictor | $\mathcal{W}$ & $filepath$ | J48 | 1387 | 79.42 | 0.94 | 0.81 |

Table 3: Overview of the best accuracies $a$ achieved by various feature sets $f$. The learning algorithm $l$, the number of used features values $g$, the micro precision $p$ and the micro recall $r$ are also given.

### 5.3 Investigating Task Detection

We now focus on investigating the following questions: (i) How good can the performance of a task classifier be when used in a real work environment, while being trained with contextual data gathered in laboratory settings? (ii) Which are the automatically observable contextual features that allow for good task detection performance? To carry out our analysis, we consider only the 203 task instances associated with Tasks 1 to 4 (see Table 2). We use the tasks recorded from the laboratory computer as the training set (106 task instances), and those recorded from the personal workstations as the test set (97 task instances). An overview of the results about the performance of detecting real workstation tasks by training on contextual data from laboratory settings is given in Table 4.

The best **feature category** is the application category, that correctly identifies 91.75% of the real tasks ($l$=NB, $g$=500, $p$=0.97, $r$=0.92). Approximately 5 points behind in terms of accuracy, is the content category ($l$=NB, $a$=86.60%, $g$=500, $p$=0.95, $r$=0.87). Using all 50 features results in a 82.47% accuracy, which is about 9% worse than the best performing feature category. This illustrates that using all available features together does not necessarily provide the best results. This is interesting because it suggests that smaller feature combinations, that

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

Table 4: Overview of the best accuracies $a$ (ranked within each section) achieved while detecting real workstation tasks by training on contextual data from laboratory settings, for each feature category, all feature categories combined, each single feature as well as the $k$ top performing single features $f$. The learning algorithm $l$, the number of used feature values $g$, the micro precision $p$, the micro recall $r$, and the global ranking $R_G$ across sections are also given.

are less computationally expensive to deal with, can achieve a better accuracy. This has directed our decision to study single features in greater details.

After evaluating the performance of each **single feature** separately, we confirm that the *window title* (Oliver et al., 2006; Shen et al., 2007; Granitzer et al., 2008) is the best discriminative feature, with an accuracy of 85.57% ($l$=J48, $g$=100, $p$=0.95, $r$=0.87). Of great interest are the good performances of our newly introduced accessibility object features: the *accessibility object name* with an 80.41% accuracy ($l$=J48, $g$=100, $p$=0.92, $r$=0.81) and the *accessibility object value* with a 71.13% accuracy ($l$=J48, $g$=150, $p$=0.89, $r$=0.72). Simply counting the number of UICO datatype relations is also quite efficient ($a$=70.10%). Seeing the good results achieved by the single features individually, we were wondering whether we could do even better by following the simple approach of combining the $k$ single features performing best with respect to classification accuracy.

We have studied the performance of the **top $k$ features**, with different values for $k$. With the *top 6* features, with the NB classifier and 250 used feature values, we obtain the highest accuracy ($a$=94.85%), precision ($p$=0.98) and recall ($r$=0.95), among all studied features, feature categories and top $k$ feature combinations. This is 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 alone. These top 6 features are: *window title*, *content in focus*, *accessibility object name*, *event-block content*, *accessibility object value*, and *UICO datatype relations*. The number of used feature values ($g$=250) of this best performing approach also supports prior work showing that a good choice for it is between 200 and 300 feature values (Shen et al., 2007).

### 5.4 Discussion

Our results only give a first impression on the fact that the working environment in which users perform their tasks has no significant influence on the task detection performance. Since only four tasks were involved in this analysis, 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 it should be true for other contextual features.

Context detection frameworks, that observe user contextual information, differ in terms of utilized sensors and of granularity of the captured contextual data. Our approach is very fine-granular. 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 deemed important, and hence captured, stored and analyzed. Using a different context detection framework could result in leaving out contextual features having a good discriminative power, and could hence have a negative impact on the task detection performance.

On the other hand, the strong positive influence of specific context features on task detection performance is an indication that it may not be necessary to track all the user's interactions with her computer desktop, but only the most relevant elements. This obviously has 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 should lead 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 requires further experiments in laboratory and real world settings.

## 6 Conclusion

We have studied the question of automatically detecting the task a user is performing on her computer desktop. We have introduced an ontology-based user interaction context model (UICO) that extends the spectrum of features that can be used for task detection. Based on these novel features we were able to provide a combination of features that outperforms other feature sets, especially those including only classical features. This result has been obtained on the dataset collected from a large user study carried out in a knowledge-intensive business environment. Our experiment has also shown that it is possible to obtain good task detection results for real user tasks with a classifier trained 'offline' on laboratory contextual data. We have also studied the discriminative power of individual and combined contextual features. The good performance of the classical *window title* feature has been confirmed and even significantly outperformed by a specific combination of 6 features. Within this combination are contextual features that are specific to our UICO approach.

The analysis presented in this paper was limited to very classical machine learning algorithms. This was a reasonable first step, as similar analysis reported in the literature use the same methods. To extend our study we now plan to use more state-of-the-art algorithms for classification and feature selection. For example, it could be interesting to use multi-class polynomial SVMs, since they allow to model feature conjunctions and dependencies. Also, instead of using the IG method for feature selection, we could use more sophisticated, multi-variate methods, such as the Recursive Feature Elimination method.

We plan to study the discriminative power of our ontology-specific contextual features more thoroughly by performing further experiments with tasks having different characteristics, since the results we have obtained here are maybe domain-specific. We have already started to

investigate this point. We have performed two similar experiments in another domain, and the results obtained confirm what is reported here (Rath et al., 2010). In general, we would like to understand why some features perform better than others for task detection. Our objective is to exhibit a small combination of contextual features with a strong discriminative power, independently of the domain, in order to enhance automatic task detection performance. More generally, since the number of controlled user studies in the task detection area is low, we plan to perform further ones to get a deeper insight on which kind of tasks can be automatically detected and in which settings. As an example, we have recently shown that knowledge-intensive tasks can be detected as well as routine tasks (Rath et al., 2010).

This work is integrated in our Knowledge Services framework *KnowSe*, which strives to provide highly contextualized and personalized knowledge services to the user. Besides providing a user task detection module, KnowSe can also perform *information need discovery*. Other services focus on *context-aware information retrieval* and *proactive context-aware information delivery*, which involve spreading activation on the graph-based representation of the user context model, and a ranking of search results based on resource usage and inter-connectivity. Among the services are also tools developed for *visualizing* the individual and organizational context of the user (graph views, time-lines and self-organizing map views).

## References


Baldauf, M., S. Dustdar, and F. Rosenberg (2007). A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing 2*(4), 263–277.

Biedert, R., S. Schwarz, and T. R. Roth-Berghofer (2008). Designing a context-sensitive dashboard for an adaptive knowledge worker assistant. In *Proc. Workshop on Modeling and Reasoning in Context, HCP '08*.

Callan, J., J. Allan, C. L. A. Clarke, S. Dumais, D. A. Evans, M. Sanderson, and C. Zhai (2007). Meeting of the MINDS: an information retrieval research agenda. *ACM SIGIR Forum 41*(2), 25–34.

Catarci, T., A. Dix, A. Katifori, G. Lepouras, and A. Poggi (2007). Task-centered information management. In *Proc. DELOS '07*, pp. 253–263.

Chernov, S., G. Demartini, E. Herder, M. Kopycki, and W. Nejdl (2008). Evaluating personal information management using an activity logs enriched desktop dataset. In *Proc. Workshop on Personal Information Management, CHI '08*.

Coutaz, J., J. L. Crowley, S. Dobson, and D. Garlan (2005). Context is key. *Communications of the ACM 48*(3), 49–53.

Dey, A. K., G. D. Abowd, and D. Salber (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human Computer Interaction 16*(2), 97–166.

Dragunov, A. N., T. G. Dietterich, K. Johnsrude, M. McLaughlin, L. Li, and J. L. Herlocker (2005). TaskTracer: a desktop environment to support multi-tasking knowledge workers. In *Proc. IUI '05*, pp. 75–82.

Dredze, M., T. Lau, and N. Kushmerick (2006). Automatically classifying emails into activities. In *Proc. IUI '06*, pp. 70–77.

Granitzer, M., M. Kröll, C. Seifert, A. S. Rath, N. Weber, O. Dietzel, and S. N. Lindstaedt (2008). Analysis of machine learning techniques for context extraction. In *ICDIM '08*, pp. 233–240.

Groza, T., S. Handschuh, K. Möller, G. Grimnes, L. Sauermann, E. Minack, C. Mesnage, M. Jazayeri, G. Reif, and R. Gudjónsdóttir (2007). The NEPOMUK project - on the way to the social semantic desktop. In *Proc. I-SEMANTICS '07*, pp. 201–211.

Gutschmidt, A., C. H. Cap, and F. W. Nerdinger (2008). Paving the path to automatic user task identification. In *proc. Workshop on Common Sense Knowledge and Goal-Oriented Interfaces, IUI '08*.

Jones, W., P. Klasnja, A. Civan, and M. L. Adcock (2008). The personal project planner: planning to organize personal information. In *Proc. CHI '08*, pp. 681–684.

Kellar, M. and C. Watters (2006). Using web browser interactions to predict task. In *WWW '06 - Poster*.

Kushmerick, N. and T. Lau (2005). Automated email activity management: an unsupervised learning approach. In *Proc. IUI '05*, pp. 67–74.

Mylonas, P., D. Vallet, P. Castells, M. Fernandez, and Y. Avrithis (2008). Personalized information retrieval based on context and ontological knowledge. *Knowledge Engineering Review 23*(1), 73–100.

Oliver, N., G. Smith, C. Thakkar, and A. C. Surendran (2006). SWISH: semantic analysis of window titles and switching history. In *Proc. IUI '06*, pp. 194–201.

Rath, A. S., D. Devaurs, and S. N. Lindstaedt (2009a). Contextualized knowledge services for personalized learner support. In *Proc. Demonstrations EC-TEL '09*.

Rath, A. S., D. Devaurs, and S. N. Lindstaedt (2009b). Detecting real user tasks by training on laboratory contextual attention metadata. In *Proc. Workshop on Exploitation of Usage and Attention Metadata, Informatik '09*.

Rath, A. S., D. Devaurs, and S. N. Lindstaedt (2009c). KnowSe: fostering user interaction context awareness. In *Proc. Demonstrations ECSCW '09*.

Rath, A. S., D. Devaurs, and S. N. Lindstaedt (2009d). UICO: an ontology-based user interaction context model for automatic task detection on the computer desktop. In *Proc. Workshop on Context Information and Ontology, ESWC '09*.

Rath, A. S., D. Devaurs, and S. N. Lindstaedt (2010). Studying the factors influencing automatic user task detection on the computer desktop. In *Proc. EC-TEL '10*, pp. 292–307.

Sauermann, L., A. Bernardi, and A. Dengel (2005). Overview and outlook on the semantic desktop. In *Proc. Workshop on the Semantic Desktop, ISWC '05*.

Sauermann, L., L. van Elst, and A. Dengel (2007). PIMO - a framework for representing personal information models. In *Proc. I-SEMANTICS '07*, pp. 270–277.

Schmidt, A. (2005). Bridging the gap between knowledge management and e-learning with context-aware corporate learning solutions. In *Proc. WM '05*, pp. 203–213.

Shen, J., L. Li, and T. G. Dietterich (2007). Real-time detection of task switches of desktop users. In *Proc. IJCAI '07*, pp. 2868–2873.

Strang, T. and C. Linnhoff-Popien (2004). A context modeling survey. In *Proc. Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp '04*.

Tang, J. C., J. Lin, J. Pierce, S. Whittaker, and C. Drews (2007). Recent shortcuts: using recent interactions to support shared activities. In *Proc. CHI '07*, pp. 1263–1272.

Van Kleek, M. and H. E. Shrobe (2007). A practical activity capture framework for personal, lifetime user modeling. In *Proc. UM '07 - Poster*, pp. 298–302.

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

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

Xiao, H. and I. F. Cruz (2005). A multi-ontology approach for personal information management. In *Proc. Workshop on the Semantic Desktop, ISWC '05*.