UICO: an ontology-based user interaction context model for automatic task detection on the computer desktop

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

to be published in Context, Information and Ontology (CIAO 2009) workshop at European Semantic Web Conference (ESWC 2009), June, 2009

UICO: An Ontology-Based User Interaction Context Model for Automatic Task Detection on the Computer Desktop

Andreas S. Rath
Know-Center GmbH
Inffeldgasse 21a
8010 Graz, Austria
+43 316 873 9265
arath@know-center.at

Didier Devaurs
Know-Center GmbH
Inffeldgasse 21a
8010 Graz, Austria
+43 316 873 9288
ddevaurs@know-center.at

Stefanie N. Lindstaedt
Knowledge Management Institute, Graz University of Technology, and
Know-Center GmbH
Inffeldgasse 21a
8010 Graz, Austria
+43 316 873 9260
slind@know-center.at

ABSTRACT

'Understanding context is vital' [1] and 'context is key' [2] signal the key interest in the context detection field. One important challenge in this area is automatically detecting the user's task because once it is known it is possible to support her better. In this paper we propose an ontology-based user interaction context model (UICO) that enhances the performance of task detection on the user's computer desktop. Starting from low-level contextual attention metadata captured from the user's desktop, we utilize rule-based, information extraction and machine learning approaches to automatically populate this user interaction context model. Furthermore we automatically derive relations between the model's entities and automatically detect the user's task. We present evaluation results of a large-scale user study we carried out in a knowledge-intensive business environment, which support our approach.

Categories and Subject Descriptors

H.1.2 [**Models and Principles**]: User/Machine Systems - *Human information processing*;
H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing - *Abstracting methods, Indexing methods*;
I.5.2 [**Pattern Recognition**]: Design Methodology - *Classifier design and evaluation, Feature evaluation and selection*

General Terms

Algorithms, Performance, Experimentation

Keywords

automatic task detection, user context model, context ontology, user context detection, machine learning

1. INTRODUCTION

Massive amounts of digital information are available to us today and are still constantly increasing. No matter if we talk about the information on the world wide web or the numerous documents, presentations, emails, and multimedia files we store on our computer desktops. Intelligent search technologies have been developed to tackle the challenge of finding and refinding information we need for achieving our goals but what is still missing is to understand the *context* in which information is used and produced. *"Understanding context is vital"* [1] and *"context is key"* [2] highlight the importance of context. As also recently discussed in the information retrieval community [3], the emphasis of future information retrieval applications ought to be put on exploiting the user's context in order to increase the accuracy of retrieval results. Detecting the user context thus appears to be of great interest for several research fields.

Two important issues in the context detection area are user context observation and user context analysis. In our previous work we have focused on context observation [4], which we briefly summarize in Section 3.1. The focus of this paper is on context analysis and more specifically on automatic task detection. Automatic task detection is an important challenge within context analysis [2, 5]: if her current task is detected automatically 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 has been so far on using only text-based features and switching sequences [6, 7, 8, 9, 10] for detecting the user's task, which do not rely on sophisticated models. Furthermore standard datasets for the evaluation of task detection approaches are still missing as well as a representative number of controlled user studies.

In this paper we propose an ontology-based user context model for increasing the performance of automatic task detection. Using an ontology-based user context model brings several advantages, such as an easy integration of new contextual attention metadata [11], 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. Most impor-

tantly we show that using an ontology-based representation of the user's context enhances the performance of automatic task detection in comparison to previous approaches. This is achieved by deriving new ontology-specific features for machine learning algorithms which increase their performance. We also present an evaluation of our approach based on a controlled user study (220 recorded task instances from 13 participants) we carried out in a knowledge-intensive business environment.

The outline of the paper is as follows: In the next section (Section 2) we describe our conceptual model and its realization as an ontology-based user context model. Section 3 elaborates on the mechanisms and techniques utilized to automatically populate the proposed ontology. In Section 4 we present the principles of our ontology-based task detection approach and how to evaluate it. Our experimental results and a comparison of our ontology-based approach to other recent task detection approaches are discussed in Section 5. Section 6 contains concluding remarks and an outlook on future work.

2. MODELING THE USER'S CONTEXT

Our view of the "*user's 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"* [5]. We specifically focus on the interactions of the user with applications and resources on the computer desktop. We refer to this focused perspective as the *user's interaction context*, or simply the user's context. Our perspective puts the individual user and her actions into the center of attention, and aims at learning as much as possible about this individual and her actions. Our goal is to study the relations between the user's interactions with the computer system and their meaning and relevance to the user's task.

Various representation formats are available for modeling the user's context. We go along with the conclusion of the surveys from Strang et al. [12] and Baldauf et al. [13] which advocate the use of ontology-based approaches. In our context specifically 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 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 will show that the performance of automatic task detection can be enhanced by using an ontology-based context model (see Section 5). This stems from the fact that our approach provides new ways of constructing features (see Section 4.1) for the machine learning algorithms, that enhance performance.

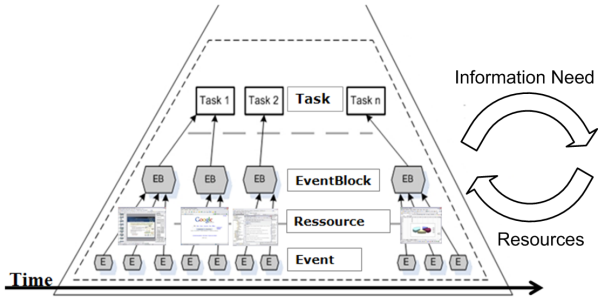


Figure 1: The semantic pyramid [4] comprises the event, the event block and the task layers. Information needs can originate from each layer and are fulfilled by resources.

2.1 Conceptual Model - The Semantic Pyramid

The user's context can be seen as a pyramid, the *semantic pyramid* (see Figure 1), which connects the user's actions with resources acted upon. At the bottom are events that result from single user interactions with the computer desktop. Above are event-blocks, which are sequences of events that belong logically together. At the top are tasks, which are well-defined steps of a process, that cannot be divided into sub-tasks, and in which only one person is involved. The layers of the semantic pyramid represent aggregation levels of user actions. It also integrates the idea of delivering resources that are relevant to the user's actions based on her information needs. For detailed definitions and a more elaborated discussion about the semantic pyramid we would like to refer to [4].

2.2 UICO: User Interaction Context Ontology

Our user interaction context ontology (UICO) can be seen as the realization of the semantic pyramid with the support of semantic technologies. We follow a bottom-up approach and build the UICO on the basis of our conceptual model, the semantic pyramid, and incrementally add relations when new sensor data or algorithms are added. UICO holds context information that has been sensed and relates the information that is automatically derived from it. By context information we mean the concepts and the relations between concepts of the semantic pyramid, as well as the resource data and metadata that have been captured by the context sensors.

The UICO's intention is to represent the user's interaction context originating from context sensors that automatically observe the user on the computer desktop and the corresponding automatically derived contextual information. If new sensor information is available new concepts and relations can be easily added to the ontology. Our methodology here is to capture and store all concepts and relations we can compute. Based on the application domain in which the UICO is used we decide which relations and concepts are useful and which are not. In the special case of ontology-based task detection we study concepts and relations that are significant for a specific task, i.e., highly discriminative between tasks. Although it would be possible to allow the

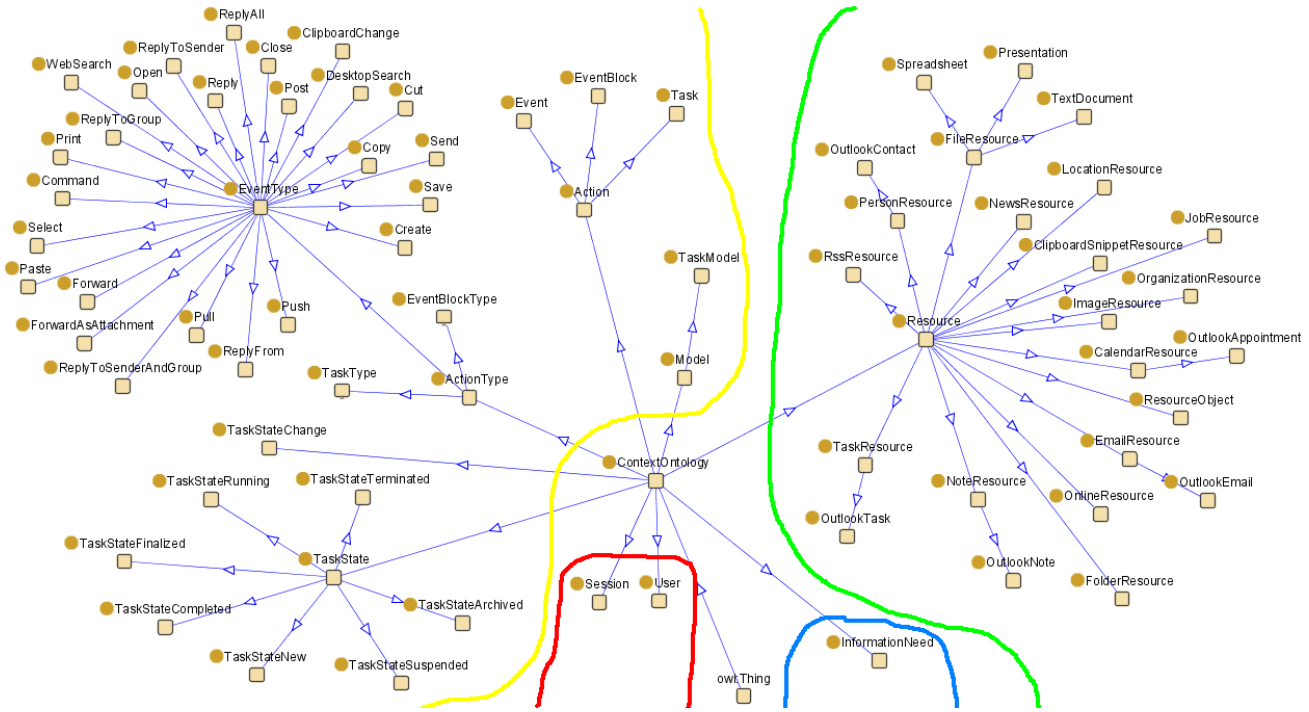


Figure 2: The concepts of the User Interaction Context Ontology (UICO) visualized in the Protégé tool. In the left area this figure shows the *action dimension*, in the right area the *resource dimension*, in the bottom left area the *user dimension* and the *information need dimension* on the bottom right area. The *application dimension* is not represented as concepts and hence is not visible here.

user to manually add new concepts and relations to the UICO this is not in focus at the moment.

At the moment UICO contains 88 concepts and 272 properties and is modeled in OWL-DL[1]. From these 272 properties there are 215 datatype properties and 57 objecttype properties. The tool used for modeling the ontology was the Protégé ontology modeling tool[2]. A visualization of the concept hierarchy in Protege is given in Figure 2.

From a top level perspective, we define in UICO five different dimensions. These are the *action dimension*, *resource dimension*, *information need dimension*, *application dimension* and *user 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 the granularity, i.e., `Event` at the lowest level, then `EventBlock` and then `Task`. These action concepts corresponds to the levels of the semantic pyramid. The `ActionType` concepts specify which types of actions are distinguished on each granularity level. Currently we only distinguish types of actions on the event level (`EventType` concept) but the elaboration of `EventBlockTypes` and `TaskTypes` is in progress. There are 25 different `EventTypes`, each one representing a single type of user interaction (see upper left part of Figure 2). As an example, if the user clicks on the search button of a search engine's web page in a web browser, this user interaction will generate an Event of type `WebSearch`.

The `TaskState` concept and its sub-concepts (visualized in the lower left part of Figure 2) are used to model the ways the user does task management and task executions. The types of task states are derived by the *Nepomuk Task Management Model* [14] and integrated into the UICO. With the help of task states UICO can model the user's task handling , i.e., creating, executing, interrupting, finishing, aborting, approving and archiving a task. The behavioral patterns of the user's task handling and task state changes are 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 in form of the `TaskModel`, a sub class of the `Model` concept, is present. Our `TaskModel` concept is similar to those defined in the area of workflow management systems or task process analysis.

At the moment, the `TaskModel` concept can be seen as a way of categorizing a task. An example of instances of the `TaskModel` and the `Task` concepts is "Planning a journey" and "Planning the journey to CIAO 2009 workshop" respectively.

[1]http://www.w3.org/2004/OWL/
[2]http://protege.stanford.edu

Resource Dimension. The resource dimension, visualized in the upper right corner of Figure 2, contains concepts for representing resources on the computer desktop. Specifically we focus on modeling resources used by knowledge-workers [15] and identified by interviews. Further resource types can be easily added if required. We define 16 different resource concepts. A resource is constructed from the data and metadata captured by the context sensors. The detailed description of the resource discovery and construction processes is given in Section 3.2.

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 are instances of the `Event` concept, connected via the objecttype property `isActionOn` to the same instance of a `TextDocument` (and a `FileResource`) representing that document.

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 context information. An `InformationNeed` concept has properties defining the accuracy of the detection and the importance to fulfill the information need in a certain time-frame. For details about information need detection we refer to [16].

An information need is associated with the user's action(s) that trigger(s) a rule. Hence a connection between the information need dimension and the action dimension exists. The resource dimension is also connected to the information need dimension, since each resource that helps for fulfilling the user's information need is related via the objecttype property `suggestsResource` with the `InformationNeed`.

User Dimension. The user dimension contains two concepts, the `User` and the `Session` concepts. The `User` concept defines basic user information such as user name, password, first name and second name. The user dimension is related to the action dimension in such a way that each `Action` is associated with a `User` via the objecttype relation `hasUser`. Indirectly the user dimension is also related to the resource dimension and the information need dimension via the action dimension. The `Session` concept is used for tracking the time of user logins and the duration of a user session in our application.

Application Dimension. The application dimension is a *"hidden"* dimension because it is not modeled as concepts in the UICO. This dimension is present is such a way that each user interaction happens within the focus of a certain application, e.g., the user's desktop, Microsoft Word or the Microsoft Windows Explorer. The `Event` concept holds the information about the user interaction with the application by 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 have GUI elements such as the window itself, scroll bar(s) and buttons for minimizing, maximizing and closing the application. Most of GUI elements have an associated *accessibility object*[3] which can be accessed by context sensors. Datatype properties of the `Event` concept hold the data about the interactions with GUI elements. Later on we show that these accessibility objects play an important role in task detection.

A resource is normally accessed by the user within an application hence there is 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 the `Event` concept.

2.3 Related Work

The UICO is similar to the *Personal Information Model Ontology* (PIMO) [17] developed in the research project NEPOMUK[4] in terms of representation of desktop resources. However, for our purposes of automatic context capturing, a limitation of the PIMO is the coarse granularity of concepts and relations. Our UICO is a fine-grained ontology, driven by the goal of representing automatically captured low-level contextual information whereas the intension of PIMO is to enable 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[5] which is used in the Mymory project [18] models native operations (e.g., `AddBookmark` or `CopyFile`) on *generic information objects* (e.g., email, bookmark or file) which are 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 than we do for the UICO's `Resource` concepts.

In [19] a layered and semantic ontology-based framework for personal information management (PIM) which follows the principles of *semantic data organization*, *flexible data manipulation* and *rich visualization* is proposed. The framework consists of an *application*, *domain* and *resource layer* as well as a *personal information space*. The resource dimension of our UICO can be seen as a combination of the domain and resource layer because resource instances are mapped to concepts of the domain layer. The intention of the approach is to propose a framework for PIM whereas the UICO focuses on representing the user interaction context. The main differences are that in the UICO there are (1) no pre-defined ontology for resources and (2) no concepts or relations for representing user actions.

3. AUTOMATIC MODEL POPULATION

For a user it is not convenient to manually enter 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

[3]Microsoft Active Accessibility:
http://msdn.microsoft.com/en-us/accessibility/
[4]http://nepomuk.semanticdesktop.org
[5]http://usercontext.opendfki.de/wiki/NopOntology

ontology. Rule-based, information extraction and machine learning approaches are utilized to automatically populate the ontology and automatically derive relations between the model's entities. We describe in this section how we build instances of concepts and augment relations between the concept instances. We show which kind of sensors we use to observe user interactions with the computer desktop, how to discover resources the user has utilized, unveil connections between resources, and aggregate single user interactions (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. Low-level operating system and application events initiated by the user while interacting with her desktop, are recorded by context observers, which is similar to the approach followed by contextual attention metadata [11] and other context observation approaches [9, 20, 21].

The data about the occurred events is sent as an XML stream to the context capturing framework for processing and analysis. Our targeted domain of the contextual attention metadata collection is the Microsoft Windows XP or Vista environment. Especially, we focus on supporting applications that knowledge workers are using in their daily work. We have identified the following applications to be worthwhile for utilization: the Microsoft Office 2003/2007 suite (Word, Excel, PowerPoint, Outlook), Microsoft Internet Explorer, Novell GroupWise, Mozilla Thunderbird and Mozilla Firefox. Context observers, also referred to as context sensors, are programs, macros and plug-ins that provide the functionality to observe the user interaction behavior on the computer desktop. We distinguish sensors based on the origin of the sensor data they deliver and talk about *system* and *application* sensors. We refer to [4] for a complete listing of the available system and application sensors and for a description of what kind of contextual information they are able to sense.

3.2 Resource Discovery

The resources that populate the ontology model are for example links, documents, persons, emails, files, folders, web pages, organizations, locations, files, folders, presentations, text documents or spreadsheets. Resource discovery is about the identification of resources and the extraction of related metadata in the contextual attention metadata stream, referred to as the *event stream*. Furthermore it is 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., copy of a part of text from an email to a text document. A resource is referenced by another resource if the location of the resource appears in the content of another resource, e.g., a link to a web page appears in the content of an email. The resources that are identified by resource discovery mechanisms are related to instances of the `Event` concept by the `isActionOn` objecttype property.

Three techniques are applied to discover resources: the *regular expression*, the *information extraction* and the *direct resource identification* approaches. (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 and email addresses for example. (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 context information. As an example, when the name of a person is identified in a text document, it is mapped to an instance of an already existing `Person` concept and a relation specifying that this person is mentioned in that document is added. (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 sensor data about an email that the user has opened. In this case the sensor sends the information that a specific email identified by the `server message id` has been opened for reading. Additional metadata about the email is attached by the sensor and added to the discovered resource. Another example is the `ClipboardSnippetResource` which is built from the content of the clipboard application sensed by the clipboard observer.

3.3 Event to Event Block Aggregation

Context sensors observe low-level context attention metadata that result in 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 in this sense means grouping the events that capture the interaction of a user with a single resource into an event block. 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. This is the case for example 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 mechanism of the application 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 id for a resource in an application then heuristics are used for uniquely identifying that resource.

Two types of rules can be distinguished for the event to event block grouping process. The first type is a set of rules designed for specific applications, and referred to as *application specific rules*. An example of such a rule is: group 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 if there is a lack of information in the event stream for applying an application specific rule. The goal of these rules is to heuristically group events into event blocks based on event attributes which can be observed operating-system-wide by the context sensors. These attributes are the window title of the application, the

process number (id) and the id of the window handle[6]. The window title and the process id perform the best for a generic event to event block grouping in which no application specific attributes are present. This discriminative power of the window title has been observed also in other work: In SWISH [6] the window title results in an approximately 70% accurate task detection. This finding goes hand in hand with Shen et al.'s task detection research [7] in which a combination of the window title and the file path of the currently edited document has been utilized. In Granitzer et al. [10] the window title is also mentioned as a good discriminating context feature for tasks.

3.4 Tasks

The aggregation of user actions into tasks is different from the previous rule-based approach since it would require to manually design rules for each task. This might be a reasonable approach for well-structured tasks, like administrative or routine tasks, but is obviously not appropriate for tasks that involve a certain freedom and creativity in the execution, e.g., for knowledge-intensive tasks like "Planning a journey" or "Writing a research paper". To be able to also handle such kind of unstructured tasks the idea 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. ONTOLOGY-BASED TASK DETECTION

A classical approach for automatic task detection is to model it as a machine learning problem (classification task). This approach has been used for recognizing web based tasks [22, 23], tasks within emails [24, 25] or from the complete user's desktop [6, 7, 9, 10]. All these approaches are based on the following steps. First, the contextual attention metadata has to be captured by context sensors. Second, it has to be chosen which parts of the data (features) are used for building the training instances for the machine learning part. Since these features can not directly be used as inputs for machine learning algorithms the third step is to transform the context features into attributes [26]. This transformation may also include data preprocessing operations. An example for the famous window title feature would be a summarization of the words appearing in the window title into a "bag of words" then transformed into vector format. For text based features, preprocessing steps like removing stopwords [7, 9, 10] or application specific terms [6], are applied. The fourth step is to apply feature selection algorithms [10, 27] to select the most important features for the learning algorithms (optional). The fifth step is the training and testing of the learned model.

In the machine learning part we use the machine learning toolkit Weka [26] for parts of the data preprocessing, filtering, feature selection, and classification.

4.1 Training Instance Construction

Constructing training instances for the machine learning algorithms is done on the task level. This means that 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 results in a *multi-class* classification problem. A training instance for a class is built from features and feature combinations of the context of an instance of a `Task` concept. The process of constructing features representing a task instance and of transforming them into attributes that can be used to train machine learning algorithms is referred to as *feature engineering*.

The context features we chose for building the training instances can be grouped in four categories: ontology structure, content, resource, and action. The *ontology structure category* contains features representing the number of instances of concepts and the number of datatype and object-type relations used per task. The *content category* stands for the set of features that involve text based content: the content of the resource displayed, the content in focus and the text input of the user. The *resource category* includes the complete contents and URIs 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* can be seen as the category that represents the user interaction. It contains features about the interaction with applications, graphical user interface elements (accessibility objects), 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.

Besides the ontology structure category, which is obviously specific to our ontology-based approach, we have also new ontology-based features in the action and resource categories. These new features are constructed based on combinations of concepts with concepts as well as concepts with concept instances. An example for the first one is the combination of the `EventType` with the sub-concepts of the `Resource` concept. For the second one the combination of the `EventType` with an instance of the `TextDocument` concept is an example.

4.2 Data Preprocessing and Feature to Attribute Transformation

Following steps were performed to preprocess the content of text-based features (in this sequence): (i) remove end of line characters, (ii) remove markups, e.g., `\&lg` and `![CDATA`, (iii) remove all characters but letters, (iv) remove German and English stopwords, (v) remove words shorter than three characters. We transformed text-based features into vectors of words representing this feature with the `StringToWordVector` function of Weka. For numeric features we applied the Weka `PKIDiscretize` filter to replace discrete values by intervals. We used the *Information Gain (IG)* measure to rank features by their discriminative power.

4.3 Evaluation

In our experiment we evaluated the influence of three parameters on the task detection performance: (i) the number of features, (ii) the classification model and (iii) the feature category. The Weka toolkit and the WEKA integration of the libSVM provided us with the tool set to study the performance of the Naive Bayes (NB), Linear Support Vector Machine (SVM), J48 decision tree (J48) and k-Nearest Neighbor (KNN-k) with $k \in \{1, 5, 10, 35\}$ algorithm.

[6]The window handle id is a unique identifier of the window constructed by the Microsoft Windows operating system.

Table 1: Overview of the best accuracies (a) for the features (f) from the UICO, Dyonipos, SWISH and TaskPredictor approaches. The learning algorihtm (l), the number of attributes (g), the micro precision (p) and the micro recall (r) are also given.

Set	f	l	g	a	p	r
UICO	Action Cat.	J48	776	83.64	0.95	0.82
	Content Cat.	NB	1409	65.00	0.87	0.67
	Resource Cat.	NB	1849	67.27	0.89	0.72
	Ontology St. Cat.	J48	232	65.00	0.87	0.63
	All Categories	**J48**	**4201**	**89.55**	**0.97**	**0.90**
Dyonipos	$ApplicationName(\mathcal{A})$	J48	18	46.81	0.78	0.50
	$Content(\mathcal{C})$	KNN-1	109	62.72	0.86	0.63
	$WindowTitle(\mathcal{W})$	J48	241	79.01	0.93	0.78
	$\mathcal{AC}$	J48	2353	69.55	0.89	0.70
	$\mathcal{AW}$	J48	313	80.91	0.94	0.81
	$\mathcal{CW}$	NB	1509	81.82	0.95	0.84
	$\mathcal{ACW}$	NB	755	82.73	0.95	0.85
SWISH	$\mathcal{W}$	J48	589	79.55	0.93	0.79
TaskPredictor	$\mathcal{W}\&filepath$	J48	701	81.82	0.94	0.82

For each classifier/learning algorithm $l \in L$ and each feature category $f \in F$ we selected the g attributes having the highest IG value to obtain our dataset. As values for g we used 50 different measure points that were equally distributed over the available number of attributes. Since we measured no performance increase by using more than 10000 attributes we introduced an upper bound at this point for the measuring interval. The total number of attributes depends on the chosen features. Stratified 10-fold cross-validation was applied and statistical values for each fold were computed. In addition, the mean and standard deviation of all values were calculated across all folds. We measured the accuracy (a) of the used algorithms, the number of attributes (g), the micro precision (p) and micro recall (r).

5. EXPERIMENT

This section describes the first results achieved from a task detection experiment we have carried out in the knowledge-intensive domain of the Know-Center. Furthermore we compare the performance of our approach to these of the TaskPredictor [7], the SWISH [6] and the Dyonipos [10] system.

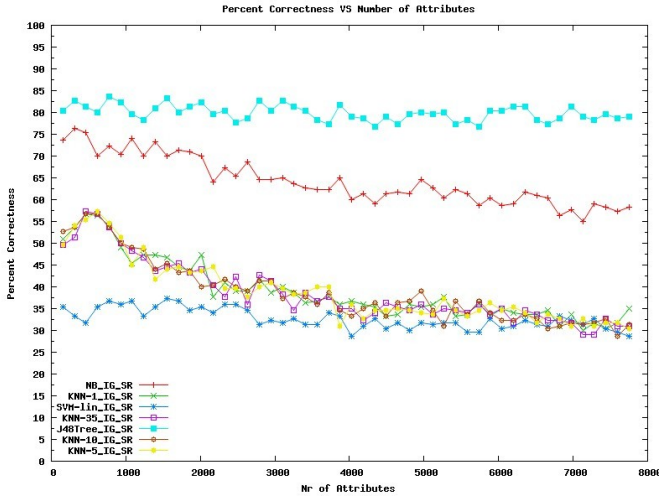
For the experiment we selected five typical task models from our domain for the experiment. Three task models were routine tasks (Task 1: "Filling in the official journey form", Task 2: "Filling in the cost recompense form for the official journey", Task 3: "Creating and handing in an application for leave") and two were knowledge-intensive tasks (Task 4: "Planning an official journey", Task 5: "Organizing a project meeting"). Before we started the experiment we did an online questionnaire to confirm that the descriptions of the task models are clear to the users and that the routine task are tasks that have already been executed by the employees several times before. The dataset we gained by the methods and techniques presented in the previous sections has 220 task instances (Task 1: 55, Task 2: 47, Task 3: 51, Task 4: 52, Task 5: 15) from 13 participants. Half of the task instances were collected on a single laboratory computer common to all users and the other half on the employees' own work station computers. The dataset contains over 1 million triples. The average number of triples per task is 4900 with a standard deviation of 4200 triples. The task with the maximum number of triples contains 29700 triples.

Figure 3 shows the performance of the seven studied learning algorithms for different numbers of attributes based on the feature combinations for the presented categories. The highest three accuracy values were achieved by the J48 decision tree algorithm with 89.55% on 4201 attributes (p=0.97, r=0.90), with 88.18% on 3801 attributes (p=0.97, r=0.88) and with 87.73% on 401 attributes (p=0.96, r=0.88). The k-Nearest Neighbor ($k = 5$) achieves 71.81% on 1201 attributes (p=0.91, r=0.74). The results which are summarized in Table 1 highlights that the J48 decision tree and the Naive Bayes algorithm perform best and that features of the action category achieve the highest accuracy compared to the other feature categories.

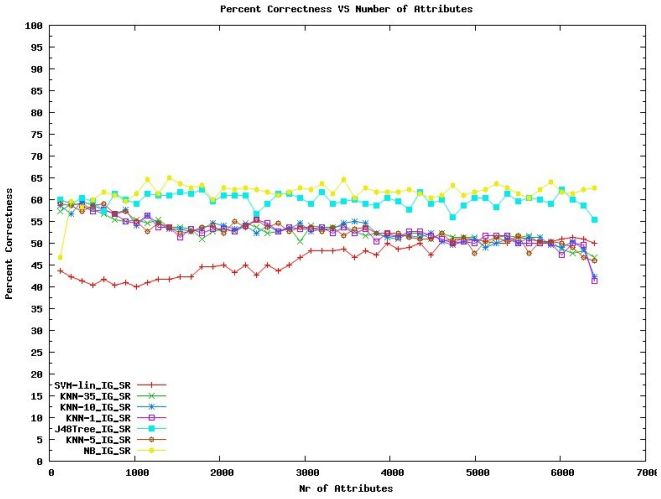
The preliminary analysis of the task detection performance of singular context features showed that the *name of the accessibility object* (l=J48, g=26, a=84.55, p=0.95, r=0.84) and the *value of the accessibility object* (l=J48, g=1765, a=72.27, p=0.90, r=0.73) are good discriminative features.

5.1 Comparison with other Task Detection Approaches

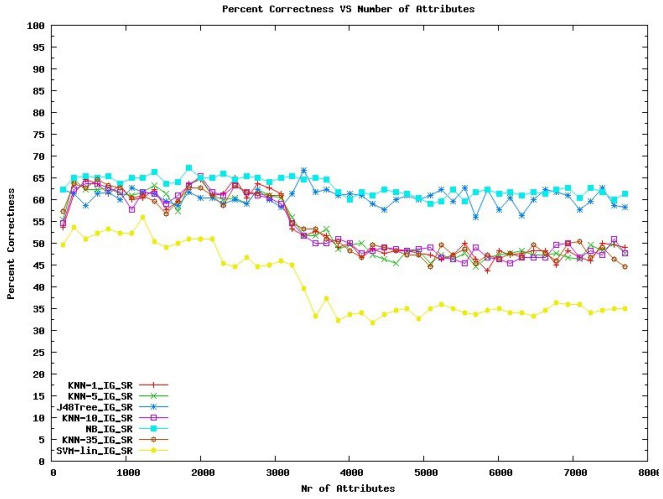
Comparing our approach to the TaskPredictor, SWISH and Dyonipos approaches can not be done directly because the construction of the training instances varies in terms of the granularity. In SWISH *window switching algorithms* are introduced that determine the boundary of a training instance, in TaskPredictor *Window-Document Segments (WDSs)* are built to make a prediction. In the Dyonipos approach one training instance per event block is constructed. In our ap-



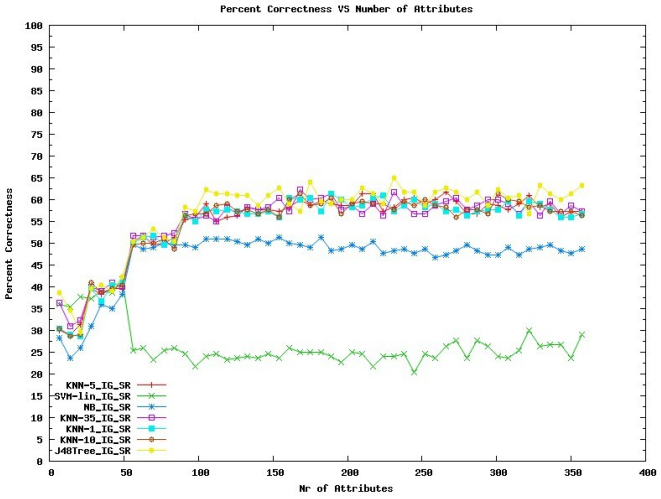
(a) Action Category



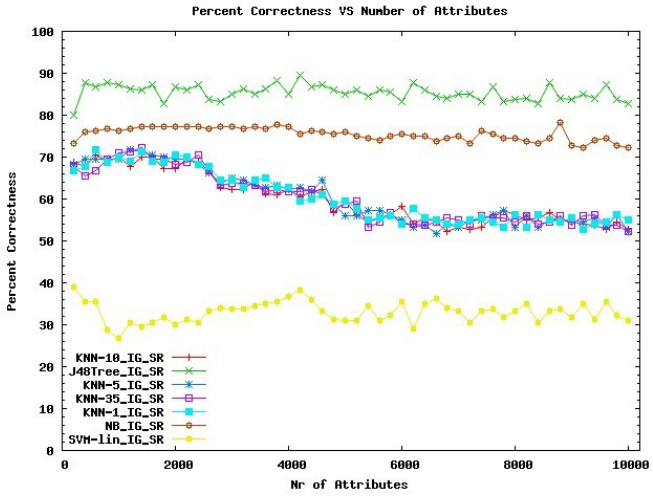
(b) Content Category



(c) Resource Category



(d) Ontology Structure Category



(e) All Categories

Figure 3: Accuracy of the learning algorithms for correctly identifying the task model a task belongs to. In the top four pictures (a)-(d) the performances of the algorithms for each category is shown. The bottom picture (e) visualizes that the combination of all features of all categories outperform each single category. The information gain (IG) feature selection was used to select the top g number of attributes visualized on the x-axis.

proach, we construct one training instance for each task. Since the number of training instances per class has an influence on the accuracy of the classification we focused on the feature engineering part for the comparison. We took the context features used by the mentioned approaches, preprocessed them as published in [6] for SWISH, [7] for TaskPredictor and [10] for Dyonipos, and evaluated the performance according to our experiment's setup (dataset and algorithms). Table 1 presents a comparison of the best algorithm runs in terms of accuracy for task detection for the various features/feature categories. It shows that the combination of the feature categories of our ontology-based approach outperforms the features used in other approaches in our experiment. The accuracy is increased by 6.82%, the micro precision by 0.02 and the micro recall by 0.05 in comparison to the best values achieved by other approaches.

For a detailed comparison of the precision and recall values between the TaskPredictor, SWISH and Dyonipos approaches we refer to [10].

6. CONCLUSIONS AND FUTURE WORK

In the digital information age with the high amount of digital information available to us it is important to know the task of the user in order to support her better. For detecting the user's task we introduced an ontology-based user interaction context model (UICO) that extends the spectrum of feature construction for automatic task detection. Based on our novel features we could outperform existing task detection approaches which we evaluated on the dataset collected from a large scale user study in a knowledge-intensive business environment.

The promising results achieved sparked interest in further investigating which features in particular are responsible for the high task detection performance. We will study combinations of the best features to find a good balance between the number of features required, the task detection speed as well as the accuracy, precision and recall of the detection. The 50 measure points interval we used to evaluate our approach only shows a tendency of the number of attributes that should be used in the classification hence we will continue to investigate finer intervals. Since the number of representative laboratory studies in the area of task detection is low we plan to do further ones to get a deeper insight on which kind of tasks can automatically be detected. The work presented in this paper will be integrated into our service-oriented knowledge services framework (*KnowSe*) which strives to provide highly contextualized and personalized knowledge services to the user.

7. 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.

8. REFERENCES

[1] S. Greenberg, Context as a dynamic construct, Human-Computer Interaction 16 (2) (2001) 257–268.

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

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

[4] A. S. Rath, N. Weber, M. Kröll, M. Granitzer, O. Dietzel, S. N. Lindstaedt, Context-aware knowledge services, in: Workshop on Personal Information Management, CHI '08, Florence, Italy, 2008.

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

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

[7] J. Shen, L. Li, T. G. Dietterich, Real-time detection of task switches of desktop users, in: IJCAI '07, Hyderabad, India, 2007, pp. 2868–2873.

[8] S. Chernov, Task detection for activity-based desktop search, in: SIGIR '08 - Doctoral consortiums, Singapore, 2008, p. 894.

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

[10] M. Granitzer, M. Kröll, C. Seifert, A. S. Rath, N. Weber, O. Dietzel, S. N. Lindstaedt, Analysis of machine learning techniques for context extraction, in: ICDIM '08, London, UK, 2008, pp. 233–240.

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

[12] T. Strang, C. Linnhoff-Popien, A context modeling survey, in: Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp '04, Nottingham, England, 2004.

[13] M. Baldauf, S. Dustdar, F. Rosenberg, A survey on context-aware systems, Int. J. Ad Hoc and Ubiquitous Computing 2 (4) (2007) 263–277.

[14] T. Groza, S. Handschuh, K. Möller, G. Grimnes, L. Sauermann, E. Minack, C. Mesnage, M. Jazayeri, G. Reif, R. Gudjónsdóttir, The NEPOMUK project - on the way to the social semantic desktop, in: I-SEMANTICS '07, Graz, Austria, 2007, pp. 201–211.

[15] P. F. Drucker, Post-Capitalist Society, HarperBusiness, 1993.

[16] A. S. Rath, M. Kröll, S. N. Lindstaedt, M. Granitzer, Low-level event relationship discovery for knowledge work support, in: Workshop on Productive Knowledge Work - Management and Technological Challenges, WM '07, Potsdam, Germany, 2007.

[17] L. Sauermann, L. van Elst, A. Dengel, PIMO - a framework for representing personal information

models, in: I-SEMANTICS '07, Graz, Austria, 2007, pp. 270–277.

[18] R. Biedert, S. Schwarz, T. R. Roth-Berghofer, Designing a context-sensitive dashboard for an adaptive knowledge worker assistant, in: Workshop on Modeling and Reasoning in Context, HCP '08, Delft, The Netherlands, 2008.

[19] H. Xiao, I. F. Cruz, A multi-ontology approach for personal information management, in: Workshop on The Semantic Desktop - Next Generation Personal Information Management and Collaboration Infrastructure, ISWC '05, Galway, Ireland, 2005.

[20] A. N. Dragunov, T. G. Dietterich, K. Johnsrude, M. McLaughlin, L. Li, J. L. Herlocker, TaskTracer: a desktop environment to support multi-tasking knowledge workers, in: IUI '05, San Diego, California, USA, 2005, pp. 75–82.

[21] M. Van Kleek, H. E. Shrobe, A practical activity capture framework for personal, lifetime user modeling, in: UM '07 - Poster, Corfu, Greece, 2007, pp. 298–302.

[22] A. Gutschmidt, C. H. Cap, 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.

[23] M. Kellar, C. Watters, Using web browser interactions to predict task, in: WWW '06 - Poster, Edinburgh, Scotland, 2006, pp. 843–844.

[24] N. Kushmerick, T. Lau, Automated email activity management: an unsupervised learning approach, in: IUI '05, San Diego, USA, 2005, pp. 67–74.

[25] M. Dredze, T. Lau, N. Kushmerick, Automatically classifying emails into activities, in: IUI '06, Sydney, Australia, 2006, pp. 70–77.

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

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