Studying the factors influencing automatic user task detection on the computer desktop

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

Studying the Factors Influencing Automatic User Task Detection on the Computer Desktop

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

[1] Know-Center GmbH., Inffeldgasse 21A, 8010 Graz
{arath,slind}@know-center.at
http://www.know-center.at
[2] CNRS; LAAS; 7 avenue du colonel Roche, F-31077 Toulouse, France and
Université de Toulouse; UPS, INSA, INP, ISAE; LAAS; F-31077 Toulouse, France
devaurs@laas.fr
http://www.laas.fr
[3] Knowledge Management Institute,
Graz University of Technology, Inffeldgasse 21A, 8010 Graz
http://www.kmi.tugraz.at.at

Abstract. Supporting learning activities during work has gained momentum for organizations since work-integrated learning (WIL) has been shown to increase productivity of knowledge workers. WIL aims at fostering learning at the workplace, during work, for enhancing task performance. A key challenge for enabling task-specific, contextualized, personalized learning and work support is to automatically detect the user's task. In this paper we utilize our ontology-based user task detection approach for studying the factors influencing task detection performance. We describe three laboratory experiments we have performed in two domains including over 40 users and more than 500 recorded task executions. The insights gained from our evaluation are: (i) the J48 decision tree and Naïve Bayes classifiers perform best, (ii) six features can be isolated, which provide good classification accuracy, (iii) knowledge-intensive tasks can be classified as well as routine tasks and (iv) a classifier trained by experts on standardized tasks can be used to classify users' personal tasks.

1 Introduction

Learning activities frequently occur within work processes [5]. The *work-integrated learning (WIL)* paradigm [14, 23] takes these observations seriously and sees learning as a dimension of work. One goal of WIL is to foster learning during work in order to enhance task performance. For assisting the learner in this kind of learning situation her *user profile* [6, 25], her interests [7], her competencies [11], and her *user context* [12, 19, 27] are utilized to improve the quality of support mechanisms. If her current task is automatically detected, the user can be supported with task specific learning and work material such as the retrieval of learning objects [4] or suggestions on course material, links, documents, or topical experts [12]. Hence it is important to know and understand what the user is working on or is trying to achieve. The user context includes all information that can be used to characterize the user's current situation [2] which also includes the user's current task. Automatic user task detection is an important challenge in

the area of user context detection [18]. The classical approach is to model task detection as a machine learning problem. However, the main focus has been so far on using only text-based features and switching sequences [3, 8, 9, 15, 16, 21, 22], which do not rely on ontology-based user context models. A recent exception is our *ontology-based user task detection* approach [18] for which we have already shown that it improves the task detection accuracy compared to the existing task detection approaches SWISH [16], Dyonipos [8] and TaskPredictor 1 [22].

In this paper, we utilize our ontology-based user task detection approach in order to unveil which features and classifiers are showing a high automatic task detection performance across three independent datasets. More specifically, our main objective is to study the factors influencing the performance of task detection: (i) the used learning algorithm, (ii) the features chosen for constructing the training instances, (iii) the kind of tasks to be classified and (iv) the method chosen for training the learning algorithms. Confronted with the lack of standard datasets and of controlled user studies in the task detection field, we have designed and run three laboratory experiments with multiple users from two different domains, for collecting task detection datasets.

Studying the influence of the type of tasks to be classified is important for increasing our understanding of the *"task detection phenomenon"* itself. Can any kind of task be classified? Can we expect similar performance when classifying routine tasks and knowledge-intensive tasks? Studying the influence of the training method on the performance of classifiers can help to address the classical *"cold start problem"*. Can a classifier, trained on standard tasks performed by experts, be used to classify users' personal tasks? The importance of studying the influence of the learning algorithm and of the chosen features lies in the fact that, in a productive scenario, it is not possible to use all available algorithms and features, because of the resulting computational cost. Our goal is to find a combination of features and classifiers that achieves good results on a standard desktop computer.

The rest of the paper is organized as follows. First, we outline our approach for recording and modeling the user context, based on our user interaction context ontology. Second, we present our ontology-based user task detection approach and our methodology for evaluating its performance. Third, we describe our three experiments and the gathered datasets. Fourth, we detail the results of our evaluation, based on the influencing factors previously mentioned. Finally, we draw our conclusions and present some future work.

2 User Interaction Context Detection

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"* [2]. We refine Dey's perspective by focusing on the *user interaction context* that we define as *"all interactions of the user with resources, applications and the operating system on the computer desktop"* [18].

A model is needed for storing 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 [24]. However, the ontology-based approach has been advocated as being the most promising one [1, 24] mainly because of its dynamicity, expressiveness and extensibility. We have defined a user interaction context ontology (UICO) [18] which represents the user interaction context through 88 concepts, 215 datatype and 57 objecttype properties. It is modeled in the *ontology web language* (OWL)[4], a W3C standard for modeling ontologies widely accepted in the Semantic Web community. The majority of concepts represents the types of user interactions and the types of resources. The high number of datatype properties represent data and metadata about resources and application user interface elements the user interacted with. The objecttype properties relate (i) the user interactions with resources, (ii) resources with other resources or parts of resources and (iii) user interactions with themselves for modeling the aggregation of user interactions. The highly connected UICO is therefore naturally enabling both: (i) a variety of context-aware applications and (ii) *"mining"* activities for in-depth analyzes of user characteristics, actions, preferences, interests, goals, etc. For a detailed description of the UICO we refer to [17, 18].

2.1 Automatic Population of the User Interaction Context Ontology

Context observation mechanisms are used to capture the behavior of the user 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. The data about the occurred events is then sent as an XML stream to the context capturing framework for discovering resources and for aggregating events (single user interactions) to event blocks (continuous sequence of user interactions on the same resource). This is similar to the contextual attention metadata approach [27] and to context observation in general [15, 18, 21, 22].

Context observers, also referred to as context sensors, are programs, macros and plug-ins that record the user's interactions on the computer desktop. We developed a broad range of context sensors for standard office applications and the operating system Microsoft Windows (XP, Vista and 7). A complete list of sensors is given in [17]. The sensed contextual data sent by the context sensors is used as a basis for automatically populating the UICO. Automatic population here means an autonomous instantiation of concepts and creation of properties between concept instances of the UICO based on the observed and the automatically inferred user interaction context. The automatic population exploits the structure of user interface elements of standard office applications and preserves data types and relationships through a combination of rule-based, information extraction and supervised learning techniques. We also use our knowledge discovery framework, the KnowMiner [10] to perform named entity recognition of persons, locations and organizations as well as for extracting data and metadata of various resource types. Hence, the UICO is a much richer representation of the user interaction context than is typically stored in attention metadata sensor streams [27] since it preserves relationships that otherwise are lost.

[4] http://www.w3.org/2004/OWL/

This rich representation of the user interaction context is exploited for machine-learning based task detection as described in the next section. At this point we would like to note that a rule-based aggregation of user actions into tasks might be a reasonable approach for highly-structured tasks, such as administrative or routine tasks, but this is obviously not appropriate for tasks that involve a certain freedom and creativity in their execution, e.g. for knowledge-intensive tasks such as "Planning a journey" or "Writing a research paper". To handle such unstructured tasks the idea is to automatically extract a task from the information available in the ontology by means of machine learning techniques. Once detected, these tasks will also populate the ontology.

3 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 switch detection involves predicting when the user switches from one task to another [16, 21]. Task classification deals with the challenge of classifying usage data from user task execution into task classes or task types. Automatic task detection is classically modeled as a machine learning problem, and more precisely a classification problem. This method is used to recognize Web based tasks [9], tasks within emails [3, 22] or tasks from the complete user's computer desktop [8, 15, 16, 21, 22].

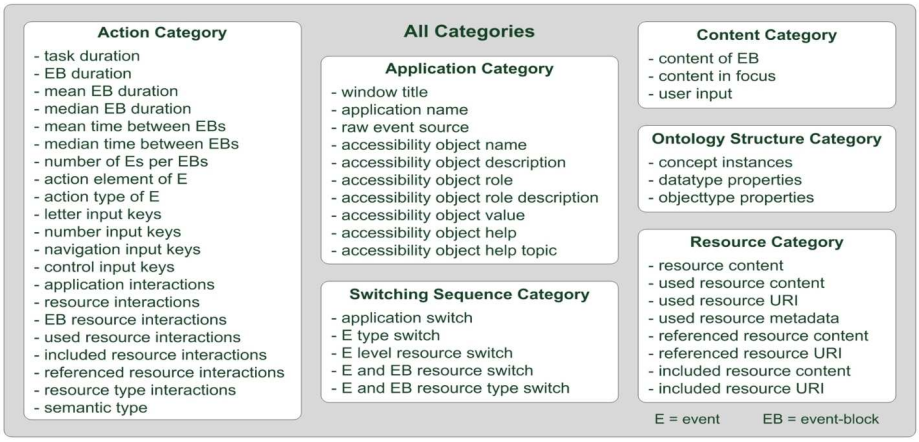
Classically, solving this classification problem is based on the following steps: (i) The user context data is captured by system and application sensors. (ii) Features, i.e. parts of this data, are chosen to build classification training instances, which is done at the task level. (iii) To obtain valid inputs for machine learning algorithms, these features are first transformed into attributes [26]. This transformation may include data preprocessing operations, such as removing stopwords [8, 15] and application specific terms [16], or constructing word vectors. (iv) Attribute selection [8, 22] (optional step) is performed to select the best discriminative attributes. (v) Finally, classification/learning algorithms are trained and tested.

Beyond this well-accepted procedure, two major limitations within the user task detection field still have to be addressed. First, the focus of the previously mentioned approaches so far has been on using only text-based features and switching sequences, which do not rely on sophisticated models. Second, standard datasets for the evaluation of task detection approaches are still missing, as well as a representative number of controlled user studies. We address this point in the next section. Regarding the first limitation, it has been recently shown that using an ontology-based context model can increase the performance of automatic task detection [18]. This new approach has been named *ontology-based user task detection*. We extend this work, and study in detail the influence of using an ontology-based user interaction context model on task detection.

3.1 Ontology-Based User Task Detection

As mentioned previously, performing task detection consists in training machine learning algorithms on classes corresponding to task models. This means that each training instance presented to the machine learning algorithms represents a task that has to

Fig. 1. Overview of all features (and their respective feature categories) engineered from our User Interaction Context Ontology (UICO).



be "labeled". Thus, training instances have to be built from features and feature combinations derived from the user context data at the task level. In our ontology-based approach, this means deriving features from the data associated with a `Task` concept.

Based on our UICO, we have engineered 50 features for constructing the training instances (see Figure 1). They are grouped in six categories: (i) *action*, (ii) *application*, (iii) *content*, (iv) *ontology structure*, (v) *resource* and (vi) *switching sequences*. The *action category* represents the user interactions and contains features about the interactions with applications [8], resources types, resources, key input types (navigational keys, letters, numbers), the number of events (E) and event blocks (EB), the duration of the event blocks, and the time intervals between event blocks. The *application category* contains the classical *window title* feature [8, 15, 16, 21], the application name feature [8] and the newly introduced accessibility objects features. The *content category* consists of features representing the content of task-related resources, the content in focus and the text input of the user. 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 *resource category* includes the complete contents and URIs [22] 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 *switching sequences category* comprises features about switches between applications, resources, event types and resource types.

We use the machine learning toolkit Weka [26] for parts of the feature engineering and classification processes. 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, e.g. `\&lg` and `![CDATA`, (iii) remove all characters but letters, (iv) remove German and English stopwords, (v) remove words shorter than three characters. For numeric features, we apply the Weka `PKIDiscretize` filter to replace discrete

values by intervals. We transform text-based features into vectors of words with the `StringToWordVector` function of Weka.

3.2 Methodology for Performance Evaluation

In all our task detection experiments, we use learning algorithms considered as classical in the text classification area. Besides, we study their performance with the same evaluation methodology, to ensure the comparability of results across the different experiment's datasets. In all our experiments, the following parameters are varied in order to evaluate their influence on the task detection performance: (i) the learning algorithm, (ii) the set of used features and (iii) the number of attributes generated from the features. Furthermore, 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 $k \in \{2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20\}$.

The evaluated learning algorithms are: Naïve Bayes (NB), Linear Support Vector Machine (SVM) with cost parameter $c \in \{2^{-5}, 2^{-3}, 2^{-1}, 2^0, 2^1, 2^3, 2^5, 2^8, 2^{10}\}$, J48 decision tree (J48), and k-Nearest Neighbor (KNN-k) with $k \in \{1, 5, 10, 35\}$. The different values for the number of neighbors k are introduced in order to explore the task detection performance with different decision boundaries. The values of the cost parameter for the SVM are borrowed from the libSVM practical guide[5]. The Weka machine learning library [26] and the Weka integration of the libSVM[6] provide the necessary tool set to evaluate these algorithms based on standard classification evaluation measures [26], such as accuracy, precision, recall and f1-measure.

For each learning algorithm $l \in L$, each feature category $\phi \in \Phi$ and each feature $f \in F$, the g attributes having the highest *Information Gain* value (IG) are selected. Information gain attribute selection is used because (i) it is one of the fastest and most popular algorithms in the text classification field, and (ii) "pre-evaluations" with more advanced attribute selection methods showed little improvement. As values for g, 50 different measure points are used for analyzing the required number of attributes for high task detection accuracy values. Half of them are equally distributed over the available number of attributes 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 attributes available, whichever is less. The choice of these specific measuring points is motivated by two reasons. First, previous evaluations of task detection performance reported that good results were achieved with a low number of attributes (about 200-300) [8, 15, 22], and hence we put a special focus on similar numbers of attributes. Second, we also want to investigate the influence of using higher numbers of attributes on task classification.

Two methods are used for evaluating the learning algorithms. First, a stratified 10-fold cross-validation is performed: statistical values for each fold are computed, as well as the mean and standard deviation of each value across the folds. Second, a training and test set evaluation is performed. The training and test sets are constructed based on

[5] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

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

the specific research question investigated in each experiment (see next section). The training set and test set instances are strictly parted (i.e. constructed and preprocessed independently) to avoid any bias. Each learning algorithm is then trained on the training set and evaluated on the test set.

We use two different techniques for isolating the best features or feature combinations, as well as the best classifiers: (i) *dominance matrices* and (ii) *paired t-tests*. Similar cross-datasets comparison methods have been proposed for evaluating learning algorithms on text categorization [28] and for evaluating hierarchical clustering algorithms on multiple document datasets [29]. The feature (resp. classifier) dominance matrix computes how often a feature or feature combination (resp. a classifier) outperforms another one. We say that a feature/feature combination (resp. classifier) outperforms another one if one of the following conditions (tested in this order) is satisfied: (i) higher accuracy (ii) higher micro precision, (iii) higher micro recall, (iv) lower number of attributes. We perform paired t-tests to study the statistic significance of the results achieved by the classifiers, based on: (i) their accuracy and (ii) the micro f-measures. We perform the paired t-tests at three different significance levels ($\alpha = 0.005$, $\alpha = 0.01$ and $\alpha = 0.5$) using the *Apache Commons Mathematics Library*[7].

4 Experiments

4.1 Related Work and Datasets

In *SWISH* [16] about four hours of real usage data observed from a single user was recorded, which gave five different tasks. In Dyonipos [8] about 140 tasks were collected from one user. The dataset used for evaluating TaskPredictor 1 [22] contains 177 tasks performed by two users, and the one used for evaluating TaskPredictor 2 [21] contains 304 tasks performed by two users (user 1: 299 tasks, user 2: 5 tasks). These datasets show several limitations for studying task detection and drawing conclusions about the performance of classifiers and features. Besides the small number of users, and the fact that each dataset relates to only one domain, a more critical issue is that no task classes were defined prior to the data collection and that the gathered tasks were labeled only afterwards and not by the user. An exception is an experiment performed in the APOSDLE project [15], in which business task classes such as *market analysis*, *product design and specification*, *find and contact suppliers*, *contract placement* and *triggering production* were predefined. But again this experiment was limited to one user and one domain. This is because of the poorness of these datasets that we decided to perform several experiments in laboratory settings, with multiple users from different domains, and with several predefined task classes. The experiments are only briefly described here because of space limitations. Further details about the design and settings of the experiments can be found in [17].

4.2 First Laboratory Study

The initial objective of *Task Experiment 1* was to study the influence on task detection of the computer environment in which tasks were performed. Users had to execute sim-

[7] http://commons.apache.org/math/

ilar tasks both in a controlled environment, namely on a single laboratory computer, and on their personal workstations. 14 probands, within the working domain of the Know-Center GmbH, participated in the experiment, which produced 218 tasks. Additional tasks were performed by one of the users, who was playing the role of the expert, which increased the total number of tasks to 271. The idea was to also evaluate the performance of a classifier trained only on tasks performed by the expert. The experiment was exploratory, the comparison was *within subjects* and the manipulations were targeting (i) the computer environment, (ii) the type of task (standard vs. personal and routine vs. knowledge-intensive) and (iii) the task class.

The first manipulation was achieved by varying the work environment, i.e. the computer desktop environment in which the participants performed their tasks. The first environment was a *laboratory computer* on which a set of standard software used in the company was installed. The second one was the company's *personal workstations* of the users, with their personal desktop settings and with access to their personal files, folders, bookmarks, emails, etc. All participants performed the same set of tasks in both environments, but half of them started on the laboratory computer, and the others started on their personal workstations. All assignments of the users were randomized.

The second manipulation was based on the task type. The two dichotomies we introduced between task types were: (i) *routine task* vs. *knowledge-intensive task* and (ii) *standard task* vs. *personal task*. A "standard task" is a task executed by the user on behalf of a *persona* (i.e. an artificial person) that we named "Bill Adams", as opposed to a "personal task" which is performed by the user for herself. All participants performed each task both in a standard and in a personal way, and the fact of starting with the standard or the personal task was randomly chosen.

The third manipulation resulted from the choice of studying five task classes, chosen by the users as being typical of their domain. Three of them were considered as **routine tasks** (*Filling in the official journey form* (55 tasks), *Filling in the cost recompense form* (45 tasks), *Creating an application for leave* (51 tasks)) and the other two as **knowledge-intensive tasks** (*Planning an official journey* (52 tasks), *Organizing a project meeting* (15 tasks)). The figures in the brackets represent the number of task executions for each task recorded in the experiment. The order in which the users had to execute the tasks was randomly generated.

4.3 Second Laboratory Study

Task Experiment 2 was designed similarly to *Task Experiment 1*, but was performed in another domain and with different task classes. 10 probands from the Computer Science Department of the Graz University of Technology participated in the experiment, which produced 134 tasks. Seven task classes were studied which resulted in the following recorded dataset: four were considered as **routine tasks** (*Registering for an exam* (19 tasks), *Finding course dates* (20 tasks), *Reserving a book from the university library* (20 tasks), *Registering for a course* (17 tasks)) and the other three as **knowledge-intensive tasks** (*Programming an algorithm* (20 tasks), *Preparing a scientific talk* (19 tasks), *Planning a study trip* (19 tasks)).

4.4 Third Laboratory Study

Task Experiment 3 was designed to study the possibility of classifying knowledge-intensive tasks and to evaluate the influence of the task type (analytic or synthetic) on task detection. The task classes we used were borrowed from the *CommonKADS* knowledge-intensive task classification [20]. 18 probands from the Computer Science Department of the Graz University of Technology participated in the experiment, which produced 132 tasks. The experiment was *within subjects*. The first manipulation was achieved by varying the type of the knowledge-intensive task, namely *analytic task* or *synthetic task*, as defined by the *CommonKADS* task classification. The second manipulation was performed by varying the subtypes of analytic and synthetic task classes. Recording the user interaction context for these task classes led to the following dataset:

Analytic tasks: *Classification*: "Classify a list of computer science terms to hardware and software" (17 tasks), *Diagnosis*: "Find the origin of a malfunction in a computer program" (15 tasks), *Assessment*: "Assess whether a student has to pay tuition fees based on her/his application" (19 tasks), *Prediction*: "Predict the questions of an exam based on historical data" (16 tasks).

Synthetic tasks: *Design*: "Create a simple conceptual design for the software of an elevator system" (18 tasks), *Assignment*: "Assign students to study groups led by study assistants" (16 tasks), *Planning*: "Plan a software project for the development of a document management system" (16 tasks), *Scheduling*: "Create a schedule for the design and development of an electronic library book lending system" (15 tasks).

5 Evaluation of Task Detection Performance

Globally over the three datasets, the different task instances could be classified with a high accuracy, by using the stratified 10-fold cross-validation method. The levels of accuracy achieved were 88.55% for Dataset 1, 94.84% for Dataset 2 and 86.43% for Dataset 3 (see Table 1). However it is important to note that these high accuracy levels were reached by finding, for each dataset, the best task detection setting among all possible settings, obtained by varying the classifier, the set of used features and the number of attributes. Even though the best results were generally achieved by considering the set of all 50 features or a combination of the best performing features, there was not one setting that could perform best across experiments. Exploring all possible settings for reaching high accuracy levels is obviously not realistic within a productive scenario. Thus, with the idea of trying to reduce the space of possible task detection settings in mind, we will now analyze individually each factor that can have an influence on task detection.

5.1 Influence of the Task Type

Standard Tasks vs. Personal Tasks A "standard task" is a task executed by the user on behalf of a *persona*, as opposed to a "personal task" which is performed by the user for herself. By having several users executing the same standard task, very similar task instances were expected, all those instances having a common *specific goal* (e.g.

Table 1. Overview of all task detection performance results of the three laboratory experiments.

Evaluations	**Exp. 1**	**Exp. 2**	**Exp. 3**
Stratified 10-Fold Cross-Validation			
Detection of the Task Model (5/7/8 Classes)	88.55%	94.84%	86.43%
Routine vs. Knowledge-Intensive Tasks (2 Classes)	94.94%	100.00%	-
Routine Tasks (4 Classes)	-	94.64%	-
Knowledge-Intensive Tasks (3 Classes)	-	100.00%	-
Standard Tasks (5/7 Classes)	88.41%	98.57%	-
Personal Tasks (5/7 Classes)	86.00%	94.05%	-
Analytic vs. Synthetic Tasks (2 Classes)	-	-	94.73%
Analytic Tasks (4 Classes)	-	-	97.14%
Synthetic (4 Classes)	-	-	85.24%
Train/Test Set Evaluation			
Personal Tasks based on Standard Task (5/7 Classes)	77.14%	92.42%	-

"Planning the trip of Bill Adams to EC-TEL 2010") contrary to personal tasks. Thus, it could seem easier to detect standard tasks than personal tasks. We evaluated this in Task Experiments 1 and 2, by trying to classify standard task instances (among 5 task classes) or personal task instances (among 7 task classes) independently, using the stratified 10-fold cross-validation method. We found only a small difference in the achieved accuracy values, of about 3.5% in favor of standard tasks.

In Task Experiments 1 and 2, we also evaluated the performance of task detection while training the classifiers on standard task instances and testing them on personal task instances. We obtained the accuracy values of 77.14% and 92.42% respectively, with 5 and 7 task classes respectively involved. These results suggest that training on task instances sharing a common specific goal is sufficient for classifying personal task instances. These results are also strongly supported by the facts that (i) the datasets of Tasks Experiments 1 and 2 included 218 task instances (113 standard / 105 personal) and 134 task instances (68 standard / 66 personal) respectively, (ii) Tasks Experiments 1 and 2 involved 14 and 10 users respectively and (iii) these experiments were performed in two different domains. The conclusion we can draw from this result is that a classifier trained by a group of experts on standard tasks performs well while classifying personal tasks performed by users.

Routine Tasks vs. Knowledge-Intensive Tasks In Experiments 1 and 2, we investigated the possibility of distinguishing routine tasks from knowledge-intensive tasks. We evaluated our learning algorithms on this two-class classification problem, by using the stratified 10-fold cross-validation method. We reached the accuracy levels of 94.94% and 100% for Tasks Experiments 1 and 2 respectively. This shows that task instances could easily be classified between routine tasks and knowledge-intensive tasks.

It could seem easier to detect routine tasks than knowledge-intensive tasks, since the latter involve more freedom and should produce very different task instances. We

evaluated this in Task Experiment 2, by trying to classify routine task instances (among 4 task classes) or knowledge-intensive task instances (among 3 task classes) independently, using the stratified 10-fold cross-validation method. The accuracy achieved was of 94.64% for routine tasks and 100% for knowledge-intensive tasks. However this last result is not well supported for two reasons: (i) this was only a three-class classification problem and (ii) the task classes involved were too different such that it was too easy for the classifiers to find distinguishing features to train on. Even though the accuracy value itself is controversial, this result shows that knowledge-intensive tasks could be detected as well as routine tasks.

Analytic Tasks vs. Synthetic Tasks Task Experiment 3 was designed to understand better what kinds of knowledge-intensive tasks could be detected. We used the dichotomy between analytic and synthetic tasks, as defined in the *CommonKADS* classification for knowledge-intensive tasks [20]. We first studied the two-class classification problem consisting in distinguishing analytic tasks from synthetic tasks, by using the stratified 10-fold cross-validation method. We obtained an accuracy of 94.73%, showing that these two classes could easily be distinguished.

We also studied the detectability of various task classes of each type (analytic vs. synthetic), by trying to classify analytic task instances (among 4 task classes) or synthetic task instances (among 4 task classes) independently, using the stratified 10-fold cross-validation method. The accuracy achieved was 97.14% for analytic tasks and 85.24% for synthetic tasks. Again this shows that knowledge-intensive tasks can be well classified by our approach.

5.2 Influence of Context Features and Feature Categories

Based on the evaluations performed on our three datasets, we can study the stability of the performance achieved by each feature and each feature category (see Table 1). By computing a dominance matrix for each experiment (based on how often a feature/feature category outperforms the others) a ranking of the features/feature categories can be obtained. An overview of the results, for the top 22 features/feature categories is presented in Table 2. Those are the features/feature categories that appear in the top 15 ranking produced by at least one dataset.

Several interesting insights are provided by Table 2. First, the good representation of features and feature categories engineered based on our ontology clearly signals the positive influence on the task detection performance of adopting our UICO approach. Second, the best results are achieved by the *Application Category* and by the combination of all 50 features (*All Categories*). The fact that the *Application Category* performs slightly better also shows that it is not true that: “The more features are considered, the better the achieved classification accuracy is”. Third, the single features achieving the best results are the *accessibility object name* and the *window title*. Besides, the standard deviation of the *accessibility object name* feature is one of the lowest, which indicates the good stability of its performance across datasets. The fact that the *accessibility object name* feature, which is specific to our UICO approach, performs slightly better than the well-known *window title* feature also signals the benefits of making use of the features derived from the accessibility objects. Accessibility objects are associated with

Table 2. Computation of the ranking of the features and feature categories. The global ranking R_G is given by the average μ_R of the rankings of the features/feature categories for all the task detection evaluations shown in Table 1 based on the three datasets (R_1, R_2 and R_3) and by the standard deviation δ_R in case of a draw.

$\mathbf{R_G}$	**Feature / Feature Category**	R_1	R_2	R_3	μ_R	δ_R^2	δ_R
1	Application Category	1	2	2	1.67	0.33	0.58
2	All Categories	3	1	1	1.67	1.33	1.15
3	accessibility object name	4	4	3	3.67	0.33	0.58
4	window title	2	3	6	3.67	4.33	2.08
5	Resource Category	6	7	4	5.67	2.33	1.53
6	used resource metadata	9	6	5	6.67	4.33	2.08
7	accessibility object value	5	12	8	8.33	12.33	3.51
8	Action Category	13	5	7	8.33	17.33	4.16
9	datatype properties	8	9	10	9.00	1.00	1.00
10	Ontology Structure Category	10	8	11	9.67	2.33	1.53
11	Switching Sequences Category	20	11	9	13.33	34.33	5.86
12	accessibility object role	15	15	15	15.00	0.00	0.00
13	resource type interactions	19	10	16	15.00	21.00	4.58
14	Content Category	7	27	12	15.33	108.33	10.41
15	application interactions	21	16	13	16.67	16.33	4.04
16	concept instances	22	14	14	16.67	21.33	4.62
17	resource interaction	31	13	15	19.67	97.33	9.87
18	content of EB	11	28	21	20.00	73.00	8.54
19	content in focus	12	30	18	20.00	84.00	9.17
20	accessibility object role description	14	25	30	23.00	67.00	8.19
21	used resource interaction	32	15	22	23.00	73.00	8.54
22	resource content	15	35	23	24.33	101.33	10.07

graphical user interface elements of applications, such as application windows and controls[8]. For example the `acc.object.name` value of the close button of a Microsoft Windows command window is "Close" and the `acc.obj.description` is "Closes the window". Fourth, if we reduce this table by considering only the features that appear in the top 15 rankings produced by the three datasets, we can isolate what we consider as being the best performing features: the *accessibility object name* feature, the *window title* feature, the *used resource metadata* feature, the *accessibility object value* feature, the *datatype properties* feature and the *accessibility object role* feature. Because of the low standard deviation values associated with them, the performance of these six features also proves to be stable across datasets. It is again worth noting that four of these features are new and specific to our UICO approach.

5.3 Influence of the Classifier

By using all the evaluations performed on our three datasets, we can analyze the stability of the performance achieved by each classifier (Naïve Bayes, Linear Support Vector Machine, J48 decision tree and k-Nearest Neighbor). By computing a dominance matrix

[8] http://msdn.microsoft.com/accessibility

Table 3. Computation of the ranking of the classifiers. The global ranking R_G is given by the average μ_R of the classifier performances for all the task detection evaluations shown in Table 1 based on the three datasets (R_1, R_2 and R_3). The standard deviation δ_R is also given.

$\mathbf{R_G}$	**Classifier**	R_1	R_2	R_3	μ_R	δ_R^2	δ_R
1	J48	2	2	1	1.67	0.33	0.51
2	NB	1	1	5	2.33	5.33	2.04
3	KNN-5	4	5	2	3.67	2.33	1.5
4	KNN-35	6	4	3	4.33	2.33	0.69
5	KNN-1	5	6	4	5.00	1.00	1.00
6	KNN-10	7	3	6	5.33	4.33	1.58
7	SVM	3	7	7	5.67	5.33	0.77

for each experiment (based on how often a classifier outperforms the others) a ranking of the classifiers can be obtained. An overview of the results is presented in Table 3. The J48 classifier obtains the first rank and the lowest standard deviation. This indicates that it is the most stable across datasets. The Naïve Bayes algorithm performs best on the first two datasets, but poorly on the third one, involving only knowledge-intensive tasks. It seems that Naïve Bayes cannot deal with the creative freedom inherent to knowledge-intensive tasks. The Linear Support Vector Machine classifier performs rather well on the first dataset, but is the worst on the two other datasets. The k-Nearest Neighbor algorithm shows rather constant results, but performs slightly better on the third dataset. Knowing that both Naïve Bayes and Linear Support Vector Machine are linear classifiers, the fact that they perform rather badly on the third dataset, contrary to J48 and k-Nearest Neighbor, might indicate that the decision boundary is non-linear in this case.

From the paired t-tests computed based on the classifiers achieved accuracy, we can derive partial orders of these classifiers, for each dataset. For Tasks Experiments 1, 2 and 3 the resulting partial orders are $\{$*J48, NB*$\} \gg \{$*KNN-1, KNN-5, KNN-10, KNN-35*$\} \gg \{$*SVM*$\}$, $\{$*NB*$\} \gg \{$*J48, KNN-1, KNN-5, KNN-10, KNN-35*$\} \gg \{$*SVM*$\}$ and $\{$*J48*$\} \gg \{$*NB, KNN-1, KNN-5, KNN-10, KNN-35*$\} \gg \{$*SVM*$\}$ with $\gg$ indicating a statistical significance on a $\alpha = 0.005$ level. The J48 decision tree and Naïve Bayes classifiers globally outperform the k-Nearest Neighbor and Linear Support Vector Machine learners, which supports the result given in Table 3. Furthermore, the paired t-tests we computed based on the micro f-measures, and which we omit here because of space limitations, are similar and confirm this result.

5.4 Comparison with Related Work

The most popular features identified for having a high discriminative power among tasks are the `window title` feature [8, 16, 22], the `file path/web page url` feature [22], and the `content in focus` feature [8]. In our findings we confirm the feature choice of these approaches and compare them to novel context features and feature categories introduced by our approach (see Figure 1).

In terms of attributes used for training the machine learning algorithms an interval of 200-300 attributes is suggested to be sufficient by [8, 22]. Our results confirm that only a small ratio of attributes are required to successfully identify tasks. The best overall accuracies were obtained on the interval between 100-500 attributes. The results that led to this interval cannot be presented here because of space limitation.

In the task detection experiments reported in [15] the SVM learning algorithm was mentioned as the one with the highest achieved accuracy. In [8] the good performance of the SVM learning algorithm was confirmed and the high accuracy achieved by the KNN learner highlighted. On our datasets the SVM showed the worst accuracy and f-measures. The good performance of the KNN learner can be confirmed. In contrast with [8] the Naïve Bayes learner performed very well across our experiment's datasets.

5.5 Discussion

The *generalizability to other tasks and domains* of the results obtained by this research work is limited because (i) only two domains with (ii) selected tasks were studied and (iii) only a sample of experts of the domain were involved in the experiments. However, this research work successfully discovered novel features and feature categories as well as classifiers that showed a stable and high task detection performance. Further experiments in other domains with other tasks and users are required in order to generalize.

The method for *finding the best possible detectability of tasks* for the UICO approach, comprising the evaluation of the feature categories, single performing features and *Top* k best performing single features, is limited from a theoretical point of view, in the sense that not all combinations of the 50 features were studied. However, from a practical point of view it is not reasonable to compute all possible feature combinations of 50 features, which would represent $2^{50} - 1 \approx 1.13 * 10^{15}$ combinations. In our research we have reduced the number of possible combinations by suggesting a set of six features. Finding a good combination of features for specific tasks of a domain is what we consider as fine tuning task classification. Besides, we believe that there is no unique feature combination performing well for all settings.

We have tested four types of classifiers: Naïve Bayes, k-Nearest Neighbors, Linear Support Vector Machines and J48 decision trees. However, *there are many more classifiers* in the area of machine learning that could prove to show a good applicability to the task detection problem.

6 Conclusion and Future Work

We have performed three laboratory experiments for evaluating the influence on our ontology-based user task detection approach of the following factors: (i) the used classifier, (ii) the selected features, (iii) the task type and (iv) the method chosen for training the classifiers. We have gained several insights from our evaluation. First, the J48 decision tree and Naïve Bayes classifiers provide a better classification accuracy than other classifiers, in our three experiments. Second, we have isolated six features that present a good discriminative power for classifying tasks, which is stable across datasets. Third, even though it could seem easier to classify routine tasks, our experiments show that

knowledge intensive tasks can be classified as well as routine tasks. Fourth, we have shown that a classifier trained by a group of experts on standardized tasks performs well while classifying personal tasks performed by users.

Our goal is to find a combination of classifiers and features that achieves good results on a standard desktop computer. In future work we will investigate in combining unsupervised learning mechanisms for identifying boundaries in the user interaction context data based on the discovered context features and applying the J48 decision tree and Naïve Bayes learning algorithms for classifying these clusters to task classes. Open questions we will address are (i) in which intervals should a clustering take place, (ii) are the context features that worked well in supervised learning also applicable in an unsupervised learning setting and (iii) develop a real-time task detection application respecting the computational power available on standard desktop computers. Accurate automatic task detection will allow a more reliable construction of fine-grained user profiles about the user's interests, competencies, learning goals and knowledge indicating events, extending what we have already shown in [13].

Acknowledgments

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

References

1. Baldauf, M., Dustdar, S., Rosenberg, F.: A survey on context-aware systems. International Journal of Ad Hoc and Ubiquitous Computing 2(4), 263–277 (2007)
2. Dey, A.K., Abowd, G.D., Salber, D.: A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Human Computer Interaction 16(2), 97–166 (2001)
3. Dredze, M., Lau, T., Kushmerick, N.: Automatically classifying emails into activities. In: Proc. IUI '06. pp. 70–77 (2006)
4. Duval, E., Hodgins, W.: A LOM research agenda. In: Proc. WWW '03. pp. 1–9 (2003)
5. Eraut, M.: Informal learning in the workplace. Studies in Continuing Education 26(2), 247–273 (2004)
6. Fischer, G.: User modeling in human-computer interaction. User Modeling and User-Adapted Interaction 11(1-2), 65–86 (2001)
7. Goecks, J., Shavlik, J.: Learning users' interests by unobtrusively observing their normal behavior. In: Proc. IUI '00. pp. 129–132 (2000)
8. Granitzer, M., Kröll, M., Seifert, C., Rath, A.S., Weber, N., Dietzel, O., Lindstaedt, S.N.: Analysis of machine learning techniques for context extraction. In: Proc. ICDIM '08. pp. 233–240 (2008)
9. Gutschmidt, A., Cap, C.H., Nerdinger, F.W.: Paving the path to automatic user task identification. In: Workshop on Common Sense Knowledge and Goal-Oriented Interfaces, IUI '08 (2008)

10. Klieber, W., Sabol, V., Muhr, M., Kern, R., Öttl, G., Granitzer, M.: Knowledge discovery using the KnowMiner framework. In: Proc. IADIS '09 (2009)
11. Ley, T., Ulbrich, A., Scheir, P., Lindstaedt, S.N., Kump, B., Albert, D.: Modelling competencies for supporting work-integrated learning in knowledge work. Journal of Knowledge Management 12(6), 31–47 (2008)
12. Lindstaedt, S.N., Ley, T., Scheir, P., Ulbrich, A.: Applying scruffy methods to enable work-integrated learning. European Journal of the Informatics Professional 9(3), 44–50 (2008)
13. Lindstaedt, S.N., Beham, G., Kump, B., Ley, T.: Getting to know your user - unobtrusive user model maintenance within work-integrated learning environments. In: Proc. EC-TEL '09. pp. 73–87 (2009)
14. Lindstaedt, S.N., Scheir, P., Lokaiczyk, R., Kump, B., Beham, G., Pammer, V.: Knowledge services for work-integrated learning. In: Proc. EC-TEL '08. pp. 234–244 (2008)
15. Lokaiczyk, R., Faatz, A., Beckhaus, A., Goertz, M.: Enhancing just-in-time e-learning through machine learning on desktop context sensors. In: Proc. CONTEXT '07. pp. 330–341 (2007)
16. Oliver, N., Smith, G., Thakkar, C., Surendran, A.C.: SWISH: semantic analysis of window titles and switching history. In: Proc. IUI '06. pp. 194–201 (2006)
17. Rath, A.S.: User Interaction Context - Studying and Enhancing Automatic User Task Detection on the Computer Desktop via an Ontology-based User Interaction Context Model. Ph.D. thesis, Graz University of Technology (2010)
18. Rath, A.S., Devaurs, D., Lindstaedt, S.N.: UICO: an ontology-based user interaction context model for automatic task detection on the computer desktop. In: Workshop on Context, Information and Ontologies, ESWC '09 (2009)
19. Schmidt, A.: Impact of context-awareness on the architecture of e-learning solutions. In: Architecture Solutions for E-Learning Systems, chap. 16, pp. 306–319. Information Science Reference, IGI Publishing (2007)
20. Schreiber, G., Akkermans, H., Anjewierden, A., Dehoog, R., Shadbolt, N., Vandevelde, W., Wielinga, B.: Knowledge Engineering and Management: The CommonKADS Methodology. The MIT Press, Cambridge, USA (1999)
21. Shen, J., Irvine, J., Bao, X., Goodman, M., Kolibaba, S., Tran, A., Carl, F., Kirschner, B., Stumpf, S., Dietterich, T.G.: Detecting and correcting user activity switches: algorithms and interfaces. In: Proc. IUI '09. pp. 117–126 (2009)
22. Shen, J., Li, L., Dietterich, T.G., Herlocker, J.L.: A hybrid learning system for recognizing user tasks from desktop activities and email messages. In: Proc. IUI '06. pp. 86–92 (2006)
23. Smith, P.J.: Workplace Learning and Flexible Delivery. Review of Educational Research 73(1), 53–88 (2003)
24. Strang, T., Linnhoff-Popien, C.: A context modeling survey. In: Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp '04 (2004)
25. Ulbrich, A., Scheir, P., Lindstaedt, S.N., Görtz, M.: A context-model for supporting work-integrated learning. In: Innovative Approaches for Learning and Knowledge Sharing. pp. 525–530. Springer (2006)
26. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, San Francisco, USA, second edn. (2005)
27. Wolpers, M., Najjar, J., Verbert, K., Duval, E.: Actual usage: the attention metadata approach. Educational Technology & Society 10(3), 106–121 (2007)
28. Yang, Y., Liu, X.: A re-examination of text categorization methods. In: Proc. SIGIR '99. pp. 42–49 (1999)
29. Zhao, Y., Karypis, G., Fayyad, U.: Hierarchical clustering algorithms for document datasets. Data Mining and Knowledge Discovery 10(2), 141–168 (2005)