Analysis of Machine Learning Techniques for Context Extraction

Michael Granitzer
Knowledge Management Institute
Graz University of Technology
mgranitzer@tugraz.at

Mark Kröll
Knowledge Management Institute
Graz University of Technology
Mark.Kroell@tugraz.at

Christin Seifert
Know-Center Graz
cseifert@know-center.at

Andreas S. Rath
Know-Center Graz
arath@know-center.at

Nicolas Weber
Know-Center Graz
nweber@know-center.at

Olivia Dietzel
m2n - consulting and development gmbh
dietzel@m2n.at

Stefanie Lindstaedt
Know-Center Graz
slind@know-center.at

Abstract

'Context is key' conveys the importance of capturing the digital environment of a knowledge worker. Knowing the user's context offers various possibilities for support, like for example enhancing information delivery or providing work guidance. Hence, user interactions have to be aggregated and mapped to predefined task categories. Without machine learning tools, such an assignment has to be done manually. The identification of suitable machine learning algorithms is necessary in order to ensure accurate and timely classification of the user's context without inducing additional workload.

This paper provides a methodology for recording user interactions and an analysis of supervised classification models, feature types and feature selection for automatically detecting the current task and context of a user. Our analysis is based on a real world data set and shows the applicability of machine learning techniques.

1 Motivation

Knowledge-intensive work plays an increasingly important role in organizations of all types. Workdays are characterized by processing and manipulating more and more digitized information. So far, indexing and search tools have been employed to make the stored information accessible. However, people realized, that systems have to take the knowledge worker's context into account to be truly supportive. As recently discussed in the information retrieval community [1], the emphasis of future information retrieval applications ought to be put on exploiting the user's current context in order to increase the accuracy of retrieval results.

Knowing the knowledge worker's current work activities, i.e., the task the user resides in, allows various ways for support. Search queries can be generated based on the user context, submitted and the results can be ranked according to the user's profile and context. Furthermore, guidance through the work process may be provided automatically and declared experts may be suggested.
Other research groups have recently started to examine approaches that are capable of identifying the current task of a user. The *TaskTracer* project [3] at Oregon State University monitors user system interactions and provides data for the machine learning system *TaskPredictor* [10]. Three major processing steps are incorporated into the system: (i) feature selection by mutual information, (ii) classification based on confidence threshold and (iii) classification with hybrid Naive Bayes and SVM classifiers. The *SWISH* [9] system, developed at Microsoft Research, approaches user task predictions by clustering windows on (i) window titles and (ii) window sequence. The basic assumption is that windows related to each other belong to the same task and that relations can be found by analyzing the window titles. Drezde et al. [4] applied various, similarity based methods to categorize emails by activities yielding promising results.

In this work, we identify machine learning techniques that are capable of task classification in a precise and timely manner. Based on recorded interaction patterns of the user, machine learning methods may be applied to match these

patterns to a predefined set of tasks representing the users current context. Otherwise, such a categorization has to be done manually. Moreover, the varying availability of computational power demands the identification of suitable ML methods for complex learning techniques might not be applicable on stand alone desktop devices any more. Furthermore, we analyze the usefulness of certain feature types. Thus, it is essential to capture these features for they are required in the classification process. Additionally, by knowing which feature types are important, more emphasis can be put on capturing them more accurately. Last but not least, we compare our findings with two other approaches in Section 4.
To sum up, we contribute our experiences to task classification including a methodology of tackling problems of this domain, characteristics of real life data and data preprocessing. Experiments were conducted taking into account different data aspects, varying feature numbers resulting from the feature selection method *Information Gain* and several classification models such as Naive Bayes, Nearest Neighbor and a Support Vector Machine (see [8] for an overview). Thus, it provides a strong foundation for further research in the field of determining the users context via inductive learning methods.

The paper is structured as follows: The methodology that we followed to create a resource for task classification is described in Sections 2 and 3. Section 2 describes acquisition and preprocessing of the data. Furthermore, the data model is introduced in this section. Experimental results are presented in Section 3. The findings are discussed and compared to two other task classification approaches in Section 4. Section 5 contains concluding remarks and an outlook on future work.

2 Data Analysis

This section describes the steps that are necessary getting from the recorded, raw data to the data that is used in our experiments and which may be used in a real world application. Therefore, the following challenges had to be considered: (i) amount of data quickly reaches a critical value, if every single click is monitored, (ii) dealing with heterogeneous data, e.g., sequence data, textual contents, nominal values, (iii) noise in the data and (iv) interaction patterns tend to be very user specific.
In order to cope with these challenges, we introduce a data model that allows data preprocessing, data aggregation and data abstraction.

2.1 Data Acquisition

Data acquisition starts on the lowest level where all application events initiated by the knowledge worker are recorded. Currently, monitored low level events are keystrokes, mouse movements and mouse clicks. Furthermore, for some applications it is possible to record the content managed by the application. In utilizing these functionality it is possible to access the content directly, when the user focuses such a supported application. Filename, author, organization, used template, document structure, whole document content, page number, window title or even just the content that the user currently sees on the screen are only small pieces of the data we ground the context construction on. For a more detailed description of the monitoring process, we refer to (*reference blinded*).

2.2 Data Model

User interactions with the system and reactions from the system to the user's interactions are represented as events. Since the events are monitored on a fine granular basis, sheer amounts of data are gathered. We introduce a data model to cope with this data mass. By continuously aggregating and combining the raw data, several levels of granularity are passed. Figure 1 illustrates the model layers ranging from events at the bottom to tasks at the top.
Subsequent events are aggregated to event blocks by using predefined static rules, which map a set of events to an event block. An example of such a static mapping can be as follows: The user opens MS Windows and writes a paragraph. This set of events is combined to an event block called *edit a word document*. Besides the semantic representation of an event block, the interior structure of an event block can be described as follows: Each event block consists of 31 different fields, 13 general and 18 application-dependent ones. It depends on the nature of an event block, i.e., which types of events it was aggregated over, whether a field contains data or not. Especially the application-dependent event blocks rarely contain data at all.
The event blocks are automatically clustered based on their content resulting in event block clusters. Event blocks that share a significant amount of words are likely to end up in the same cluster. The event block clusters are validated and labeled by the user thus becoming task instances and the ground truth for learning the users task model.

2.3 Data Description

In our experiments we used following four fields: *Application_Name, Content, Window_Title* and *Semantic_Type*. Taking only 4 out of 31 fields into account needs explanation and justification. Analysis of the field

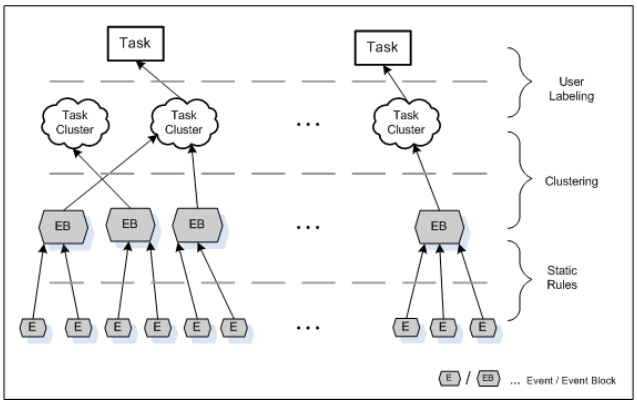

Figure 1. Relationships between events, event blocks and tasks for context detection.

characteristics showed that fields containing content are unevenly distributed. The majority of these *empty* fields are application dependent. Furthermore, we opted for those fields providing data that is homogeneously distributed over all event blocks, so to say adequately covering the entire data set. In the following we describe the four chosen fields in greater detail.

Application_Name abbreviated in following figures and tables as $\mathcal{A}$. This field contains the abbreviation of the application the user is working with. For instance, if the user reads a document in Adobe Acrobat Reader, the field will contain the value *acrord*. Sensors for following applications have been implemented: MS Word, MS PowerPoint, MS Excel, MS Internet Explorer. In addition, a file system sensor, a clipboard sensor and a generic Windows XP system sensor allow for 14 different entries in the field *Application_Name*. Entries such as *iexplor* (aka Internet Explorer) and *notepad* can be found among them.

Content abbreviated in following figures and tables as $\mathcal{C}$. In this field the content of the current window is stored. This can be the text on the current MS PowerPoint slide or the content of the website the user is viewing. Naturally, content can be written in different languages. In our data set German and English content occurs, even mixed, e.g., a webpage with German content and English navigation elements. The raw *Content* field also contains special characters, markup and end-of-line characters.

Window_Title abbreviated in following figures and tables as $\mathcal{W}$. This field holds the title of the current window. Depending on the application *Window_Title* may (MS Word) or may not (Novel Groupwise, new email window) include the name of the application.

Semantic_Type abbreviated in following figures and tables as $\mathcal{S}$. This field contains a rule-based type generated by the event collecting tools. It can take three different values, indicating whether a document is (i) read or (ii) edited by the user, or whether the state is (iii) unknown. The assignment of a semantic type is application dependent. Only MS Word, MS PowerPoint and MS Excel sensors are capable of allotting the semantic types (i) reading or (ii) writing. The discrimination is based on simple rules. If one key stroke is detected, the whole event block is classified as *writing*, otherwise as *reading*. Event blocks generated by other applications receive the semantic type *unknown*.

2.4 Data Preprocessing

We performed the following steps to preprocess the raw content of the fields (in this sequence): (i) remove EOL, (ii) remove markup, e.g., `\&lg` and `![CDATA`, (iii) remove all characters but letters, (iv) remove German and English stopwords.
As stated above the fields do not contain words from a single language, English and German text is even mixed. The application of only one stemming algorithm is not feasible any more. We tested the influence of using (i) an English stemmer and (ii) a German stemmer on the task classification performance. As we did not find significant differences by using these stemmers, we omitted stemming in further experiments.
For each event block we formed a 'Bag of Words' for all fields and calculated the TF-IDF[1] measure. We used the *Information Gain (IG)* measure to rank features by their discriminative power. Information Gain is among the most common measures for feature selection in machine learning.

2.5 Training Data Acquisition

The knowledge worker is offered several task suggestions resulting from the clustering. The assignment of event blocks can be rearranged manually by the knowledge worker. Some event blocks may be incorrectly assigned and contribute additional noise besides the noise coming from the sometimes inaccurate sensor readings. The labeling of the tasks is done by the user providing the training data for our further analysis. Noteworthy in comparison to other approaches is that knowledge workers are not bound to predefined task labels, thus admitting more flexibility in the work process. This freedom bears the risk of introducing additional noise into the data, so to say another challenge. However, the goal is to provide as much freedom for the user as possible and to take care of this noisy data on the

[1] term frequency−inverse document frequency

algorithmic level.

The user is free to assign labels as she wants. One such example is `Process Authoring an Article: Research`[2].
The prize of the freedom of unrestricted labelling is that we get a lot of syntactically different task labels from the user obviously meaning the same task (e.g *author an article* and *write an article*). We identified misspellings in the labels as well. For these two reasons we could not directly use the user's labels as task class labels. Instead, we manually identified task classes and assigned the new class labels to the corresponding event blocks. We defined two different sets of task labels. Set I is activity driven, i.e., giving priority to the own tasks at hand. Set II is process driven, i.e., a well defined step in a process, and incorporates a semantic division of the event blocks.

Set I consists of the following 5 tasks:

1. **email** - writing, reading, archiving email messages
2. **paper** - writing papers and articles
3. **research** - research to a given field
4. **documentation** - writing project documentation and protocols
5. **information** - collect information

Set II consists of the following 4 semantic tasks:

1. **com** - communication related event blocks, such as reading and writing emails
2. **org** - organizational and administrative things, such as writing timetables, make appointments
3. **write** - writing documents, articles, papers
4. **read** - reading internet pages, documents etc.

The user's task labels were manually mapped to one of the task labels by using 'human common sense'. For example, the user label `Process Authoring an Article: Research` was mapped to **research** in case of Set I and **read** in case of Set II. Event blocks whose user labels could not be sensibly mapped to a task label were omitted from further processing.
Finally, we would like to clarify, that a snapshot of one user's training data was utilized to conduct the experiments. In our real time application, user feedback is intended to continuously update the classifier.

[2]The original German labels are `Process Artikel verfassen: Recherche`

3 Experiments

In this section we describe which experiments were performed and how the task classifier was evaluated.

3.1 Experimental Data

We conducted experiments using different combinations of event block fields (see Section 2.3). Depending on the combinations, the necessary field contents are read out, concatenated to a longer string and preprocessing steps are applied. Further, the user labels are mapped to task labels according to Section 2.5 yielding Dataset I and Dataset II respectively. There are two different reasons why an event block will be removed from the data set. Firstly, no mapping from the user label to a task is found. Secondly, instances containing no data at all. Hence, field combinations differ in the number of instances. Tables 1 and 2 give an overview of the number of instances, the number of attributes (the number of different terms in the term vector) and the number of classes for selected field combinations for both datasets.

Table 1. Dataset I: Overview of the number of attributes, instances and classes for all event blocks for selected field combinations.

f	$\mathcal{A}$	$\mathcal{C}$	$\mathcal{S}$	$\mathcal{W}$	$\mathcal{ACSW}$
#ATTRIB	14	3601	4	253	3706
#INST	349	170	77	307	349
#CLASSES	5	5	5	5	5

Table 2. Dataset II: Overview of the number of attributes, instances and classes for all event blocks for selected field combinations.

f	$\mathcal{A}$	$\mathcal{C}$	$\mathcal{S}$	$\mathcal{W}$	$\mathcal{ACSW}$
#ATTRIB	14	3718	4	356	3827
#INST	406	196	77	266	406
#CLASSES	4	4	4	4	4

3.2 Experimental Design

In our experiments we evaluated the influence of three parameters: (i) the number of features, (ii) the classification model and (iii) the field combination.
We compared three classification models - Naive Bayes, Support Vector Machines and k-Nearest Neighbor. A Naive

Bayes classifier was chosen because of its simplicity and effectiveness. Furthermore, Naive Bayes relies on the global data distribution, being robust and therefore less prone to noise. SVMs, besides performing well with textual data [7], put more weight to training data close to the decision boundary. Here, a linear kernel was applied - according to [5] the corresponding feature space possesses enough discrimination power. Consequently polynomial and radial basis function kernels were left out. As representative for a non-linear, more local classifier, we opted for the nearest neighbor classifier, the number of neighbors $k \in \{1, 5, 10\}$ to achieve different complexities of the decision boundary.
The WEKA machine learning library [11] provided the necessary tools. The WEKA integration [6] of the libSVM [2] allowed the usage of a SVM as well.
The next paragraph illuminates the cross validation procedure including all possible combinations of fields, number of features and classification models. L represents the set of all classifiers, $L = \{$KNN-1, KNN-5, KNN-10, NB, SVM-lin$\}$. 15 different field combinations P were taken into account resulting from forming the power set $P \in \Pi(F) \setminus \emptyset$, where $F = \{\mathcal{A}, \mathcal{C}, \mathcal{S}, \mathcal{W}\}$. For each classifier $l \in L$ on each field combination $f \in P$ we selected the g attributes having the highest IG value to obtain our dataset. The value g was set to $g = 1, 2, 3, \ldots, 100\%$ of all available attributes (the total number of attributes depends on the field combination). Stratified 10-fold cross-validation was applied and statistical values for each fold were computed. We evaluated the accuracy of the used techniques and for comparison with related work precision/recall measures. In addition, mean and standard deviation of all values were calculated across all folds.

3.3 Results

Table 3 shows the results for the Dataset I, Table 4 for the Dataset II. For each field combination/classifier pair we selected the number of features, where the accuracy of the classifier reached a maximum. If two values coincided at maximum, the combination with fewer features was chosen. Tables 3 and 4 display the best results achieved in terms of accuracy and corresponding features for each field combination/classifier pair. Maximal performance values for each field combination are highlighted in boldface.
In Table 5 the three topmost values for Dataset I and Dataset II are listed for better comparison showing the classifier, the field combination, the mean accuracy $\hat{\mathcal{A}}$, the standard deviation of the accuracy $\sigma(\mathcal{A})$ and the number of features.

Table 5. Overview of the best results for Dataset I and Dataset II. Showing the field combination, the number of features and the classifier for the three topmost accuracy values.

SET		f	l	$\hat{\mathcal{A}}$	$\sigma(\mathcal{A})$	g
I	1	$\mathcal{SC}$	KNN-1	74.51	9.51	108
	2	$\mathcal{C}$	KNN-5	74.12	7.94	144
	3	$\mathcal{W}$	KNN-10	73.60	8.23	156
II	1	$\mathcal{W}$	KNN-1	76.42	7.90	188
	2	$\mathcal{AW}$	KNN-10	74.90	8.08	230
	3	$\mathcal{WS}$	SVM-LIN	74.74	4.45	74

4 Discussion

The results of our experiments need to be interpreted in order to be applicable in similar settings. Our analysis concentrates on determining the impact of feature selection, the field combination and the classification model.

4.1 Effect of Feature Selection

Ranking the features by discrimination capability and handing them over allows to build classification models that achieve high accuracies. The results in Table 5 reflect that only 2-6% of the total number of features sufficed for both datasets. These findings confirm, that task identification can be successfully performed using very few features.

4.2 Effect of field combinations

Each field is capable of adding information that might be valuable for separating the classes. Although the two datasets differed significantly from each other, the single fields (*Content* and *Window_Title*), that performed best at classification, coincided. Consequently, the same fields occur in field combinations that reached highest accuracies. In contrast, fields that performed poorly as individuals, e.g., the *Application_Name*, seldom boost the classification in combination with others. Considering the overall performance, we conclude that accurate classification results can be achieved by taking only the *Window_Title* into account confirming the outcomes of *SWISH*. We refer to the results in Tables 3 and 4, revealing that all classification models reached top accuracies when $\mathcal{W}$ was used.

4.3 Effect of classifier

In case of Dataset I, the victorious classification models are well-balanced. All of them succeed at least for one field

Table 3. Classifier Performance for all field combinations for Dataset I. The table shows the maximum accuracy (averaged over all cross-validation runs) and the corresponding number of selected features.

	KNN-1	KNN-5	KNN-10	NB	SVM-lin
$\mathcal{A}$	**55.06%** (7)	55.03% (5)	55.03% (5)	55.03% (5)	55.02% (5)
$\mathcal{C}$	72.35% (144)	**74.12%** (144)	73.53% (180)	65.29% (144)	69.41% (72)
$\mathcal{S}$	44.29% (3)	44.29% (3)	**44.64%** (3)	44.46% (1)	42.86% (1)
$\mathcal{W}$	73.30% (168)	73.26% (158)	**73.60%** (156)	69.68% (252)	71.95% (40)
$\mathcal{AC}$	65.29% (864)	65.59% (720)	64.49% (1224)	**65.62%** (252)	65.04% (216)
$\mathcal{AS}$	55.32% (13)	**55.35%** (10)	55.34% (6)	55.04% (4)	55.03% (5)
$\mathcal{AW}$	70.81% (250)	71.06% (164)	**71.08%** (172)	67.36% (62)	71.07% (60)
$\mathcal{SC}$	**74.51%** (108)	73.33% (108)	73.92% (72)	66.44% (144)	67.75% (108)
$\mathcal{WC}$	69.25% (2232)	69.22% (2700)	68.94% (2988)	68.62% (108)	**71.23%** (360)
$\mathcal{WS}$	72.02% (154)	**72.66%** (160)	71.97% (244)	70.39% (244)	72.39% (40)
$\mathcal{AWC}$	67.61% (1813)	67.34% (629)	67.34% (1739)	66.18% (777)	**68.79%** (259)
$\mathcal{AWS}$	69.95% (124)	69.66% (122)	69.90% (144)	63.91% (266)	**71.66%** (68)
$\mathcal{ASC}$	64.20% (1368)	64.15% (720)	64.18% (1188)	63.30% (684)	**65.92%** (252)
$\mathcal{WSC}$	70.59% (2232)	70.88% (3672)	71.18% (3312)	69.57% (288)	**71.53%** (360)
$\mathcal{AWSC}$	69.03% (3404)	**69.35%** (3256)	68.80% (3404)	66.21% (185)	69.31% (296)

combination (refer to Table 3). Dataset II, however, seems to prefer the nearest neighbor classifier (refer to Table 4). This preference leads to the assumption that the decision boundary might have become non-linear to a certain extent due to the composition of the task classes (see Section 2.5). The differing number of neighbors required could indicate the degree of non-linearity. The decision boundary tends to be complex if only one neighbor is taken into account, getting smoother in case of more neighbors. Linear SVM and Naive Bayes being linear classifiers are less capable of modeling the more complex decision boundary.

Furthermore, we noticed a relation between classification model and number of features used. Figure 2 confirms our hypothesis depicting achieved accuracies for Naive Bayes, k-Nearest Neighbors and linear SVM. The linear SVM outperforms the other approaches when few features are present and looses its prediction ability in the face of a lot of features. The k-Nearest Neighbors offer a more constant performance over the whole feature range, outperforming the linear SVM from approximately 700 features upwards. The Naive Bayes could not keep up with the accuracies the other classification models reached. Recapitulatory, the classification model is dependent on the number of features that are taken into account and consequently an adequate feature selection. In experimental settings where features are rare (in our case field combinations where the field *Content* is missing), the choice of the classification model has to be made diligently.

4.4 Comparison with other Approaches

Firstly, we address the main differences to the *TaskPredictor* project (see [10] for further detail). (i) The task labeling is restricted by offering only choices coming from a drop down box. In our setting the user was free in labeling her tasks in her own, personal way. (ii) Only window title, file path and URL are taken into account for the prediction task. (iii) They did not experiment with different numbers of features, but always used a fixed one, i.e., 200 features. (iv) The prediction results achieved a precision of 0.8 (see Table 6). It is not determinable, whether micro or macro precision was calculated. Furthermore, they introduced the term *coverage* specifying the degree of discarding certain test examples from being classified.

Secondly, we compare our approach to the *SWISH* system. The features in *SWISH* are restricted to the terms obtained from window titles. When focusing on the optimization of the recall measure, they achieved a precision of 0.49 and a recall of 0.72. In Table 6 the key data of both evaluations is compared. The number of features used was not determinable. The clusters generated in *SWISH* such as *buying a book of Harry Potter* do not seem to generalize well. The keywords of this cluster are too specific to allow an abstraction for other books. In our setting we emphasize general task classes as for instance **research** and **write**.

On the whole, we achieved significantly better preci-

[3] W- window title, P - folder path, U - URI

[4] at a coverage of 0.1

[5] at a coverage of 0.2

Table 4. Classifier Performance for all field combinations for Dataset II. The table shows the maximum accuracy (averaged over all cross-validation runs) and the corresponding number of selected features.

	KNN-1	KNN-5	KNN-10	NB	SVM-lin
$\mathcal{A}$	57.17% (3)	**57.18%** (3)	57.16% (4)	57.18% (6)	57.16% (5)
$\mathcal{C}$	72.95% (222)	73.05% (2516)	**74.03%** (2553)	67.89% (333)	65.82% (74)
$\mathcal{S}$	**74.46%** (1)	73.93% (1)	74.29% (1)	74.46% (3)	**74.46%** (1)
$\mathcal{W}$	**76.42%** (188)	75.89% (192)	76.17% (214)	70.27% (64)	74.44% (130)
$\mathcal{AC}$	65.77% (703)	65.74% (2516)	**65.79%** (518)	65.27% (37)	64.32% (148)
$\mathcal{AS}$	**57.20%** (7)	57.19% (14)	57.17% (8)	57.16% (3)	57.17% (10)
$\mathcal{AW}$	74.65% (224)	74.40% (226)	**74.90%** (230)	69.24% (250)	73.62% (108)
$\mathcal{SC}$	**72.29%** (3404)	71.24% (3404)	71.71% (3478)	70.29% (37)	66.43% (37)
$\mathcal{WC}$	71.02% (2394)	71.32% (1672)	**71.61%** (2394)	63.23% (76)	69.33% (38)
$\mathcal{WS}$	74.44% (86)	74.48% (96)	74.18% (188)	68.56% (158)	**74.74%** (74)
$\mathcal{AWC}$	**72.87%** (266)	71.70% (304)	72.42% (266)	66.75% (76)	68.69% (76)
$\mathcal{AWS}$	**74.12%** (194)	73.70% (184)	73.87% (174)	61.58% (106)	73.88% (102)
$\mathcal{ASC}$	65.28% (629)	**65.57%** (1739)	64.79% (1702)	60.38% (148)	65.02% (111)
$\mathcal{WSC}$	72.43% (2622)	**72.98%** (3154)	72.44% (1330)	62.69% (76)	68.82% (114)
$\mathcal{AWSC}$	**72.41%** (3686)	71.95% (3420)	71.98% (2622)	63.57% (114)	68.73% (190)

Table 6. Comparing *SWISH* (S), *TaskPredictor* participants FA and FB, and our approach (I, II). Showing the field combinations (f), the classification model (l), the number of tasks (t), the number of features (g) as well as the precision π and recall ρ values.

	f	l	t	g	π	ρ
I	$\mathcal{W}$	KNN	5	156	0.91	0.75
II	$\mathcal{W}$	KNN	4	188	0.90	0.74
S	$\mathcal{W}$	PLSI	5	(ALL)	0.49	0.72
FA	WPU[3]	NB/SVM	96	200	0.8[4]	N.A.
FB	WPU	NB/SVM	81	200	0.8[5]	N.A.

sion/recall values (see Table 6). Partly responsible for the performance decrease could be the differing granularity of the task classes, being more general in our case. In contrast to the other approaches, different field combinations and their influences on the task categorization were analyzed. Based on our findings, we confirm the field choice $\mathcal{W}$ of *TaskPredictor* and *SWISH*.

5 Conclusions and Future Work

Our contribution comprises three major statements that are based on a thorough analysis of the conducted experiments.

- Task classification can be accurately performed by applying machine learning techniques to captured user context.
- Our findings confirm the feature type choice of two other research groups.
- We highlighted that standard machine learning techniques suffice to extract the context in order to support the knowledge worker. Thus, stand alone desktop devices with low computing power are not deprived of this functionality.

In the presented work we use a relatively small dataset for training and evaluation. The acquisition of more data is currently in progress, recording data from several users over a period of four weeks. We are going to evaluate our methodology on this enlarged data set.

Our design allows for extending the currently available sensors, if it is required. Still, it would be interesting to add sensors that are capable of capturing telephone conversations or meeting discussions. Monitoring the physical state of knowledge workers to enrich the context by an additional dimension is not far fetched any more.

Finally, a reassessment of the feature creation process is to be considered. Instead of mixing features of all fields, a separating them in combination with different weighting schemes could be conceivable. A second approach would be to take structure into account and applying SVMs, where this additional information can be embedded into the kernel.

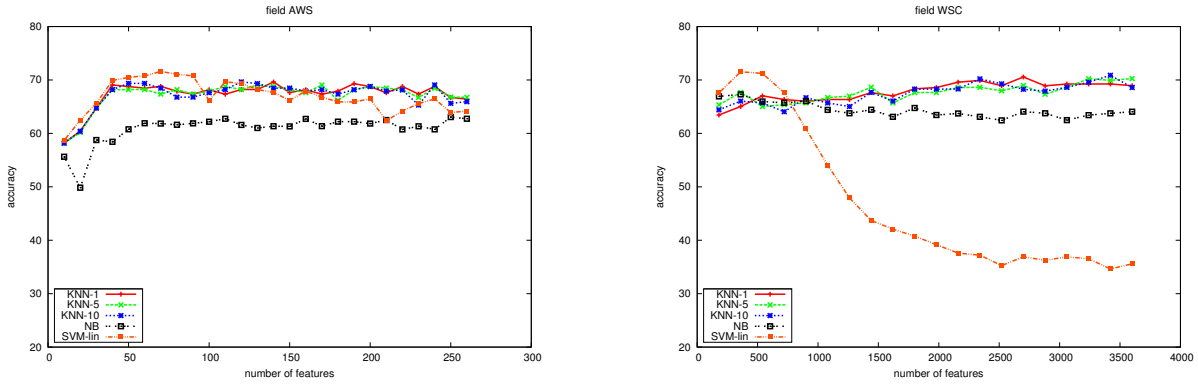

Figure 2. The performance of all three classification models varies when faced with either a lot of features or very few of them. Two field combinations $\mathcal{AWS}$ (left) and $\mathcal{WSC}$ (right) from Dataset I are considered, while Dataset II shows similar behavior.

Acknowledgment

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 (www.ffg.at). The project results have been developed in the DYONIPOS project (www.dyonipos.at) financed by the FFG under the project contract number 810804/9338.

References

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

[2] C.-C. Chang and C.-J. Lin. LIBSVM: a library for support vector machines, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[3] A. N. Dragunov, T. G. Dietterich, K. Johnsrude, M. McLaughlin, L. Li, and J. L. Herlocker. Tasktracer: A desktop environment to support multi-tasking knowledge workers. In *Proceedings of the International Conference on Intelligent User Interfaces*, pages 75–82, New York, NY, USA, 2005. ACM Press.

[4] M. Dredze, T. A. Lau, and N. Kushmerick. Automatically classifying emails into activities. In *Proceedings of the International Conference on Intelligent User Interfaces*, pages 70–77, 2006.

[5] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings International Conference on Information and Knowledge Management*, pages 148–155, New York, NY, USA, 1998. ACM Press.

[6] Y. EL-Manzalawy and V. Honavar. WLSVM: Integrating libSVM into Weka environment, 2005. Software available at `http://www.cs.iastate.edu/~yasser/wlsvm`.

[7] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In C. Nédellec and C. Rouveirol, editors, *Proceedings of the European Conference on Machine Learning*, number 1398, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[8] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[9] N. Oliver, G. Smith, C. Thakkar, and A. C. Surendran. SWISH: semantic analysis of window titles and switching history. In *Proceedings of the International Conference on Intelligent User Interfaces*, 2006.

[10] J. Shen, L. Li, T. G. Dietterich, and J. L. Herlocker. A hybrid learning system for recognizing user tasks from desktop activities and email messages. In *Proceedings of the International Conference on Intelligent User Interfaces*, pages 86–92, New York, NY, USA, 2006. ACM.

[11] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.