

Perceived Task-Difficulty Recognition from Log-file Information for the Use in Adaptive Intelligent Tutoring Systems

Ruth Janning¹ · Carlotta Schatten¹ ·
Lars Schmidt-Thieme¹

Published online: 26 May 2016

© International Artificial Intelligence in Education Society 2016

Abstract Recognising students' emotion, affect or cognition is a relatively young field and still a challenging task in the area of intelligent tutoring systems. There are several ways to use the output of these recognition tasks within the system. The approach most often mentioned in the literature is using it for giving feedback to the students. The features used for that approach can be high-level features like linguistics features which are words related to emotions or affects, taken e.g. from written or spoken inputs, or low-level features like log-file features which are created from information contained in the log-files. In this work we aim at supporting task sequencing by perceived task-difficulty recognition on low-level features easily extracted from the log-file. We analyse these features by statistical tests showing that there are statistically significant feature combinations and hence the presented features are able to describe students' perceived task-difficulty in intelligent tutoring systems. Furthermore, we apply different classification methods to the log-file features for perceived task-difficulty recognition and present a kind of higher ensemble method for improving the classification performance on the features extracted from a real data set. The presented approach outperforms classical ensemble methods and is able to improve the classification performance substantially, enabling a perceived task-difficulty recognition satisfactory enough for employing its output for components of a real system like task independent support or task sequencing.

✉ Ruth Janning
janning@ismll.uni-hildesheim.de

¹ Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Hildesheim, Germany

Keywords Adaptive intelligent tutoring systems · Task sequencing · Perceived task-difficulty recognition · Classification · Classification performance improvement · Feature analysis · Log-file features

Introduction

Nowadays, intelligent tutoring systems are important tools for supporting the education of students, for instance in learning mathematics. The main advantages of intelligent tutoring systems are the possibility for a student to practice any time and anywhere, as well as the possibility of adaptivity and individualisation for a single student. Usually, an adaptive intelligent tutoring system possesses an internal model of the student and a task sequencer that decides which tasks in which order are shown to the student. This work aims at supporting task sequencing in adaptive intelligent tutoring systems by perceived task-difficulty recognition. This perceived task-difficulty recognition is applied to information easily extracted from the log-file. More explicitly, we try to classify features extracted from the log-file as 'the student felt over-challenged by the last task' (i.e. the last task was too hard), 'the student felt under-challenged by the last task' (i.e. the last task was too easy) or 'the student felt appropriately challenged' (i.e. the last task was appropriate), that can be summarised as *perceived task-difficulty*. Originally, the task sequencing in adaptive intelligent tutoring systems was developed using information gained from expert and domain knowledge and from logged information about the performance of students in former exercises. In Schatten and Schmidt-Thieme (2014) a new efficient sequencer, based on a performance prediction system was presented, which only uses former performance information from students to sequence the tasks without the need for expensive expert and domain knowledge. This approach uses the machine learning method matrix factorization (see e.g., Cichocki et al. (2009)) to provide performance. Subsequently, it uses the output of the performance prediction process to sequence the tasks according to the theory of Vygotsky's Zone of Proximal Development (Vygotsky (1978)). That is, the sequencer chooses the next task in order to neither bore nor frustrate the student, or in other words, the next task should not be too easy nor too hard for the student. This is done by choosing a task with a predicted performance closest to a certain threshold that symbolises the performance score that the student would have, if he/she will remain within the Zone of Proximal Development. This kind of task sequencing can be supported by our perceived task-difficulty recognition by building a personalised threshold for the approach in Schatten and Schmidt-Thieme (2014) and by detecting misconceptions by means of a comparison of performance and perceived task-difficulty.

A further question, besides the questions which input features to use for perceived task-difficulty recognition and how to use perceived task-difficulty recognition within intelligent tutoring systems, is how to realise the perceived task-difficulty recognition. This means a classification approach has to be chosen. According to the literature, a state-of-the-art classification approach for affect recognition is a support vector machine (see e.g. Schuller et al. (2011)). However, usually the collection of data from young students is costly and complex and privacy issues have to be

considered carefully. These facts lead to small data sets in this area and finally to a limited performance of the support vector machine (SVM) applied to the data. Hence, the question arises, is there a way to improve the classification performance? Janning et al. (2014e) presented an approach for improving the affect recognition performance of SVMs applied to features extracted from speech. In this work we will further investigate this approach on the one hand by applying it to log-file features instead of speech features and on the other hand by applying the approach of Janning et al. (2014e) also to other classification models. The goal is to investigate which classification model performs best for the log-file features and if the above mentioned approach for performance improvement also works on classification models other than SVMs. Furthermore, in opposite to Janning et al. (2014e) where we solved a binary classification problem, in this work we address a multi-class classification problem and show that the approach of Janning et al. (2014e) also works for multi-class classification.

The main contributions of this article are: (1) presentation of log-file features for perceived task-difficulty recognition in intelligent tutoring systems, (2) statistical analysis of the presented features and experiments with different classifiers applied to the features showing the ability of the features to describe the perceived task-difficulty of a student, (3) presentation of an ensemble method for improving the perceived task-difficulty recognition performance on the presented features, and (4) proposal for the use of perceived task-difficulty recognition in intelligent tutoring systems for supporting task sequencing.

The article is structured as follows: after some related work we present the proposed log-file features on the one hand for recognising the overall perceived task-difficulty of a student and on the other hand for recognising the perceived task-difficulty of a student per task. Subsequently, we describe our approach for improving the perceived task-difficulty recognition performance for a multi-class classification problem and with different classification models. Finally, we discuss how to support task sequencing by means of the proposed perceived task-difficulty recognition approach.

Related Work

To develop an affect – or as in our case perceived task-difficulty – recognition approach appropriate features and classes have to be identified and one has to decide about the classification model to use. To recognise affect in speech linguistics features (like n-grams and bag-of-words) and low-level features like disfluencies (e.g. speech pauses) or articulation features (see e.g. Schuller et al. (2011) and Janning et al. (2014e)) can be distinguished. If linguistics features are used, a transcription or speech recognition process has to be applied to the speech input before affect recognition can be conducted. Linguistic features for affect and emotion recognition from conversational cues were presented and investigated e.g. in D’Mello et al. (2008) and D’Mello and Graesser (2012). Acoustic-prosodic features as well as lexical features are used in the context of emotion prediction in Forbes-Riley and Litman (2004), Litman and Forbes-Riley (2004) and Ai et al. (2006). Low-level features are used in

the literature for instance for expert identification, as in Worsley and Blikstein (2011), Morency et al. (2013) and Luz (2013), or for emotion and affect recognition as in Moore et al. (2014) and Janning et al. (2014e), or for humour recognition as in Purandare and Litman (2006). The advantage of using low-level features like disfluencies is that instead of a full transcription or speech recognition approach only for instance a pause identification has to be applied before computing the features. This means that one does not inherit the error of the full speech recognition approach. Furthermore, these features are independent from the need that students are using words related to affects. But still those features inherit the error from the preprocessing step. Features that are even easier to extract, which are robust against errors from preprocessing steps and which also are independent from the need that students are using words related to affects are features obtained from information about the actions of the students interacting with the system (see e.g., Mavrikis (2008)) like features extracted from a log-file (see e.g., Baker et al. (2012), San Pedro et al. (2013), Pardos et al. (2014)). In Mavrikis (2008) such features are used to predict whether a student can answer questions correctly in an intelligent learning environment without requesting help and whether a student's interaction is beneficial in terms of learning. Also in D'Mello et al. (2007) features are extracted from the tutor log-files. These features are mainly related to the dialogue of the tutor with the student, whereas the log-file features presented in this work are related to the actions of the students. The reason is that in former studies, where students were prompted to explain aloud their observations and solutions to mathematical tasks, it was observed, that the students often exhibited longer pauses of silence while thinking about the problem when they were *over-challenged* or produced a lot of short pauses while communicating when they were *appropriately challenged* (see Janning et al. (2015)). The assumption for the log-file features is that this behaviour is also observable in the interactions of the students with the system. Also the keystroke dynamics features used in Epp et al. (2011) belong to this kind of features. In Epp et al. (2011) emotional states were identified by analysing the rhythm of the typing patterns of persons on a keyboard. Because of the mentioned advantages of such features, in this work we consider features extracted from the log-file. Different from other works the proposed log-file features are created for recognising the perceived task-difficulty for supporting task sequencing. A further possibility of obtaining features is using the information from physiological sensors as in Arroyo et al. (2014), where sensors like camera and seat cushion are applied, or in Woolf et al. (2009), where a facial expression system, a posture analysis seat, a pressure mouse and a wireless skin conductance sensor are used. In Cooper et al. (2010) sensors in combination with interaction data are applied for emotion recognition. The 9 features extracted from the interaction data are similar to the log-file features presented in this work, however the features in this work are more fine granulated like the *average and maximal time distance between actions*. The reason is that we assume for instance that the longer the time distance between actions of the student the more he/she probably is thinking about the task and the more challenging is the task for the student. However, bringing sensors into classrooms is time consuming and expensive and one has potentially to cope with students' acceptance of the sensors (and the consent of their parents). Furthermore, we aim to enable

an intelligent tutoring system allowing that the student can practice anytime and anywhere, which reduces the application possibility for sensors.

Research has identified what kind of emotions or affects can be recognised in intelligent tutoring systems. In D’Mello et al. (2008) boredom, confusion, delight, flow, frustration, neutral and surprise were investigated and for boredom, confusion, flow, frustration and neutral statistically significant relationships were discovered. D’Mello et al. (2007) treats anger, boredom, confusion, contempt, curiosity, disgust, eureka, flow and frustration and it was found out that boredom, confusion, and frustration were reported at higher rates than were anger, contempt, and disgust. Woolf et al. (2009) looked at frustration, motivation/flow, confidence, boredom and fatigue. However, as mentioned above, in this work we want to use the students behavioural information gained from log-files for recognising the perceived task-difficulty and supporting the sequencing of tasks. That is, the goal is to neither bore the student with too easy tasks nor to frustrate him/her with too hard tasks, but to keep him/her in the *Zone of Proximal Development* (according to Vygotsky (1978)). Accordingly, we want to use the output of the automatic perceived task-difficulty recognition to get an answer to the question “Was this task too easy, too hard or appropriate for the student?”. With other words, we want to recognise the personalised task-difficulty from the view of the student, i.e., we want to find out if the student felt *under-challenged*, *over-challenged* or *appropriately challenged*. We use five different *perceived task-difficulty* classes for this recognition task, labelled by the students themselves (see Fig. 1): *too hard*, *hard*, *appropriate*, *easy* and *too easy*. Different to the study in Robison et al. (2009), where the answers of the participants to the question how they feel was used for an immediate feedback, in this work the self-reports of the students are used for building an automatic perceived task-difficulty recognition without the need of students directly expressing their emotions.

The possible methods for automatic emotion, affect or cognition recognition depend on the kind of features used as input. Linguistics features are obtained by a preceding speech recognition process and can be processed by methods coming from the areas sentiment classification and opinion mining (Sadegh et al. (2012)). Especially methods from the field of opinion mining on microposts seem to be appropriate if linguistics features are considered. State-of-the-art methods in opinion mining on microposts use methods based on optimisation approaches (see e.g., Hu et al. (2013)) and Naive Bayes (see e.g., Saif et al. (2012)). The process of obtaining disfluencies like pauses is different from the full speech recognition process. For extracting for instance pauses usually an energy threshold on the decibel scale is used as in Luz (2013) or an SVM is applied for pause classification on acoustic features as in Qi et al. (2004). The articulation features proposed in Janning et al. (2014e) are obtained by an intermediate step of speech recognition. For extracting features from log-files instead there is no special extraction method needed. The most important method for these features is the writing of the log-file. Appropriate state-of-the-art methods for automatic affect recognition on low-level features like disfluencies, but also on log-file features, are classification methods like SVMs, decision trees or ensembles of those (see Schuller et al. (2011) and D’Mello et al. (2008)). The best performing

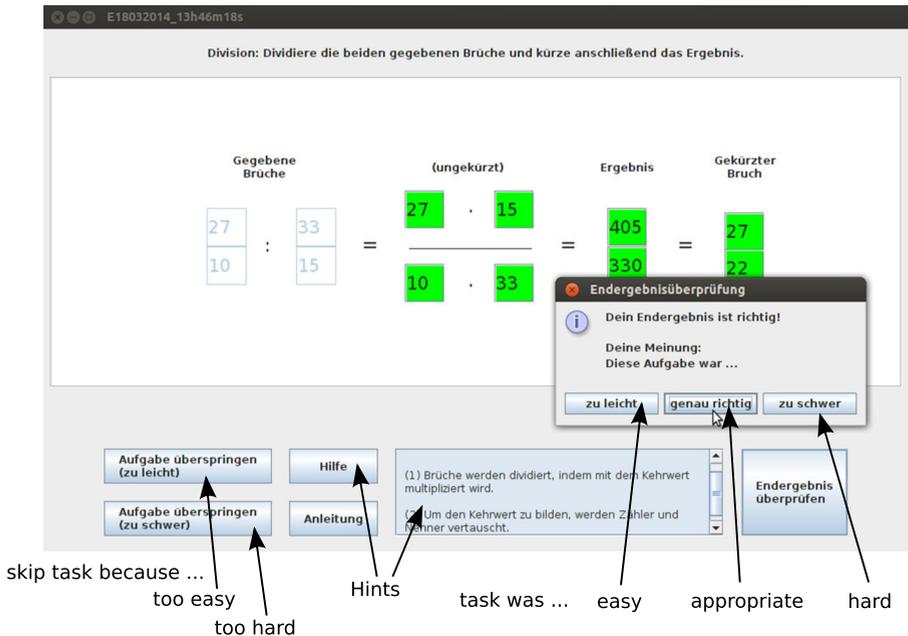


Fig. 1 Self-labelling tutoring tool

state-of-the-art method in this field is, according to the literature, an SVM. SVMs (Boser et al. (1992), Cortes and Vapnik (1995)) are supervised machine learning methods that can be used for classification tasks and deliver in many areas the best performance in comparison to other classification approaches. The library LIBSVM (Chang and Lin (2011)) provides an efficient and often used implementation of an SVM. The idea for the presented ensemble method for improving the classification performance is based on former work (Janning et al. (2013), Janning et al. (2014a), Janning et al. (2014d)), where a hybrid neural network plait (HNNP) for improving the classification performance on small and noisy signal data sets was developed and investigated. The HNNP approach uses different feature sets from different information sources and different kinds of neural networks with adapted architecture which are retrained interactively within a plait structure using additional side information gained before and during the retraining for a further improvement. The SVM plait (Janning et al. (2014e)), a first step towards the adaption of HNNP for affect recognition, has a similar structure but it uses the same kind of SVMs within the plait structure and feature subsets of one information source. The architecture of the SVMs within the SVM plait structure does not need to be adapted, the additional input is added instead to the input feature vectors. In this article we propose a generalised version of the SVM plait, which is applicable to other classification models beyond SVMs and for multi-class classification. The presented generalised plait

structure uses the principles of ensemble methods (which use multiple learning models to obtain a better prediction performance) like stacking, or stacked generalisation respectively, which are explained and investigated for instance in Ting and Witten (1999). Different from plain stacking, the presented plait consists of several layered stackings with different additional inputs.

Log-File Features

As mentioned above, we aim at using features extracted from the log-file for perceived task-difficulty recognition and supporting task sequencing. Such features are for instance the time the student spent for a task, the time between actions or the time until the first hint request or result check of the student. The idea behind this kind of features is that for instance usually the longer the time spent for a task the more challenging is the task for the student. Another example is that usually the longer the time distance between actions of the student the more he/she probably thinks about the task and the more challenging is the task for the student. The advantage of using such features is that they are very easy to obtain and there is no need of speech recognition or natural language understanding which would introduce a further source of errors. In the following, we will investigate features extracted from the log-file from two different perspectives: (1) for recognising the overall accumulated perceived task-difficulty of the student for the whole learning session (to investigate in general if those kind of features are able to describe students' perceived task-difficulty in our data), and (2) for recognising the perceived task-difficulty of a student per task. The idea behind the features for both cases is the same, but they have to be adapted a little bit for each case. The data for extracting the *log-file features* originate from a study where 7 students in the age of 21 – 28 each had to solve 40 mathematical tasks and to label at the end of a task the perceived task-difficulty themselves by means of the small offline self-labelling tutoring java tool described in Janning et al. (2014b) and shown in Fig. 1. This tool was used for the study as a drill and practice math system that does not model the student knowledge, but provides only the log-file data and the labels. The mathematical tasks were chosen from the following areas:

- Reducing fractions with numbers and variables
- Adding fractions with and without intermediate steps and with numbers and variables
- Subtracting fractions with and without intermediate steps and with numbers and variables
- Multiply fractions with and without intermediate steps and with numbers and variables
- Divide fractions with and without intermediate steps and with numbers and variables
- Distributivity law with and without intermediate steps
- Finite sums of unit fractions
- Rule of Three

The students could either solve the task and label it after solving the task with *easy*, *appropriate* or *hard* or they could skip the task and label it as *too hard* or *too easy* depending on the reason why they skipped it. Furthermore, the students could ask for several hints per task, where the last hint contains the solution of the task. Every action of a student, like pressing buttons, typing in numbers, asking for a hint or submitting the answer, was written – together with a time stamp – into a log file immediately after the action, enabling the extraction of the log-file features described below. Each of the 7 students has seen all 40 tasks resulting in 280 labelled tasks, or more explicitly 40 with label *too hard* (after skipping the task), 20 with label *hard*, 93 with label *appropriate*, 121 with label *easy* and 6 with label *too easy* (after skipping the task). For those experiments in which we tried to recognise the perceived task-difficulty of a student per task we used only examples with label *hard*, *appropriate* or *easy* (according to our goal to support the task sequencer), i.e. those examples of tasks which were not skipped.

Recognising Accumulated Perceived Task-Difficulty

Before the log-file features can be created some measurements have to be extracted from the log-file. The measurements for creating the log-file features for recognising the accumulated perceived task-difficulty of a student i are listed in Table 1 and the resulting features are shown in Table 2. These features reflect the behaviour of the student interacting with the system, since they report about the actions of a student related to task skips, time spent, hint requests, inputs and result

Table 1 Measurements for student i from the log-file input data, which are needed for extracting the desired log-file features for recognising accumulated perceived task-difficulty

Symbol	Explanation
hs_i	Number of hints seen
ha_i	Number of hints available
n_{ts_i}	Number of skipped tasks
$skip_i^{(u)}$	Time until u th skip
n_{skip_i}	Number of skips
$t_i^{(v)}$	Time spent for the v th task
$hf_i^{(w)}$	Time until first hint request for the w th task
$ht_i^{(x)}$	Time between x th and $(x + 1)$ th hint request
r_i	Number of all result checks
wr_i	Number of wrong result checks
$rf_i^{(y)}$	Time until first result check for the y th task
$wrong_i^{(z)}$	Number of wrong inputs for the z th task
$action_i^{(a)}$	Time between the a th and $(a + 1)$ th action
n_{a_i}	Number of actions

Table 2 Features created from log-file data for recognising accumulated perceived task-difficulty

Features	Formulas
Average, minimal, maximal time until skip	$\frac{\sum_u skip_i^{(u)}}{n_{skip_i}}, \min_u(skip_i^{(u)}), \max_u(skip_i^{(u)})$
Average, minimal, maximal time spent for a task	$\frac{\sum_v t_i^{(v)}}{n_{t_i}}, \min_v(t_i^{(v)}), \max_v(t_i^{(v)})$
Number of hints seen of all hints	$\frac{hs_i}{ha_i}$
Average, minimal, maximal time until first hint request	$\frac{\sum_w hf_i^{(w)}}{n_{hf_i}}, \min_w(hf_i^{(w)}), \max_w(hf_i^{(w)})$
Average, minimal, maximal time distances between hint requests	$\frac{\sum_x ht_i^{(x)}}{n_{st_i}}, \min_x(ht_i^{(x)}), \max_x(ht_i^{(x)})$
Average, maximal number of wrong inputs	$\frac{\sum_z wrong_i^{(z)}}{n_{t_i}}, \max_z(wrong_i^{(z)})$
Number of wrong result checks of all checks	$\frac{wr_i}{r_i}$
Average, minimal, maximal time until first result check	$\frac{\sum_y rf_i^{(y)}}{n_{t_i}}, \min_y(rf_i^{(y)}), \max_y(rf_i^{(y)})$
Average, maximal time distance between actions	$\frac{\sum_a action_i^{(a)}}{n_{a_i}}, \max_a action_i^{(a)}$

checks. The accumulated perceived task-differently labels per student are obtained by the following function:

$$acc. \text{ label of a student } i = rnd \left(\frac{1}{t} \sum_t \left(\begin{matrix} \text{if skip} & \begin{cases} \text{if skip because too easy} & 4.0 \\ \text{if skip because too hard} & 0.0 \end{cases} \\ \text{if not skip} & \begin{cases} \text{if easy} & 3.0 \\ \text{if ok} & 2.0 \\ \text{if hard} & 1.0 \end{cases} \end{matrix} \right) \right) \tag{1}$$

where *rnd* is a round function, *t* is the number the of tasks shown to the student and skip is a boolean variable with value 1 for the case that the student skipped the task or 0 if the student did not skip the task. To find out if the presented features are suitable to describe the perceived task-difficulty of a student we applied a statistical analysis on single features as well as on feature combinations. Hence, as a first step we investigated the relevance of each of the presented features by mapping the feature values to the accumulated perceived task-difficulty labels, doing a linear regression and measuring the p-value, indicating the statistical significance, as well as the *R*² and Adjusted *R*² value, indicating how well the regression result can approximate the real data points. The results with the best p-values are shown in Table 3. As one can see in Table 3, the feature *number of hints requested* delivers the best p-value when it is mapped to the perceived task-difficulty labels. However, as expected, most of the single features are not very significant, as for statistical significance one would desire a p-value smaller than 0.05 and values for *R*² and Adjusted *R*² that are close to 1. A more reasonable approach is to combine several features instead of considering just one feature. Hence, we also investigated different combinations of features by

Table 3 Best p-value, R^2 and Adjusted R^2 value for the log-file features mapped to accumulated perceived task-difficulty labels (reported are only features with $p < 0.12$)

Log-file feature	p-value	R^2	Adjusted R^2
Number of hints requested	0.0060	0.8075	0.7690
Maximal time between hint requests	0.0955	0.4569	0.3483
Average time between actions	0.1152	0.4205	0.3046

mapping all considered feature values to the accumulated perceived task-difficulty labels. Subsequently, we applied a multivariate linear regression to obtain the p-value, R^2 and Adjusted R^2 of the combinations. The feature combinations we started our analysis with, were obtained from generating the correlation matrix. We created a set for each feature by choosing those features for the set that are least correlated to the considered feature, i.e., with correlation values nearest to 0. In further steps we removed from these feature sets one feature after the other and applied a statistical test to each of the generated feature sets with different numbers of elements. The feature combinations with the best p-values, i.e. those with a p-value smaller than 0.05, are listed in Table 4. The best feature combination consists of 2 features (*average time until skip, average time until first hint request*). The p-value of 0.00516 of this feature combination is small enough (compared to 0.05) and the (Adjusted) R^2 value of 0.9282 (0.8923) of this feature combination is close enough to 1, to assume that this combination of features is statistically significant and hence is able to describe the behaviour of a student expressed as the accumulated perceived task-difficulty in our data. This means that it is possible to recognise the overall perceived task-difficulty of a student of our study by means of the presented features. However, as mentioned above, the accumulated perceived task-difficulty is measured per student and a relatively small amount of students participated in our study. Hence, in the next section we will investigate perceived task-difficulty recognition per task providing a much larger amount of examples to investigate and classify.

Table 4 p-value, R^2 and Adjusted R^2 for the best combinations of log-file features (with a p-value smaller than 0.05) of a set with a number (#) of 4, 3 or 2 features mapped to the perceived task-difficulty labels

#	Log-file features	p-value	R^2	Adjusted R^2
4	Maximal time between actions, maximal time spent for a task, minimal time until first hint request, maximal time between hint requests	0.00918	0.9954	0.9862
3	Average time until first result check, average time until skip, average time until first hint request	0.01543	0.9561	0.9121
2	Average time until skip, average time until first hint request	0.00516	0.9282	0.8923

Recognising Perceived Task-Difficulty per Task

The *log-file features* presented above for recognising the accumulated perceived task-difficulty per student have to be adapted to be applicable for describing students’ behaviour for single tasks instead of for the whole learning session with all tasks. More explicitly, the features describing an action, which occurs just once during one task, have to be adapted. That is *average, minimal, maximal time spent for a task* are converted to *time spent for a task* and the same has to be done with *average, minimal, maximal time until first hint request, average, maximal number of wrong inputs* and *average, minimal, maximal time until first result check. Average, minimal and maximal time until skip* instead are removed as we only consider examples of tasks that were not skipped for recognising the perceived task-difficulty per task and supporting task sequencing. The resulting features which we used in the experiments for recognising the perceived task-difficulty per task are shown in Table 5.

As mentioned above, the statistical analysis of the features per student was applied on a very small set of examples. Hence, we conducted the same analysis additionally with the features per task. The results are shown in Tables 6 and 7. In the case of single features (Table 6) the p-values are very small but unfortunately the (Adjusted) R^2 values are not very close to 1. The best feature combination (Table 7) delivers also a very small p-value and a much better result for (Adjusted) R^2 . We conducted these statistical analyses as a first step for investigating in general if the log-file features are appropriate for perceived task-difficulty recognition. For a perceived task-difficulty recognition applicable in an intelligent tutoring system, classifiers have to be applied to the features. Hence, in the following we discuss experiments with different classifiers applied to the presented features. As a first step in the classifier experiments, we applied for the multi-class classification problem (classes *hard, appropriate, easy*) an SVM, a decision tree (DCT, see e.g., Breiman et al. (1984)) and the k-Nearest-Neighbour approach (KNN, see e.g. Cover and Hart (1967)) to the features extracted from the real data set obtained in the above described study. An SVM is a classifier which searches for a hyperplane which optimally – with maximal margin – separates

Table 5 Log-file features for recognising the perceived task-difficulty per task i

Features	Formulas
f_0 : Time spent for a task i	t_i
f_1 : Number of hints seen of all hints for task i	$\frac{hs_i}{ha_i}$
f_2 : Time until first hint request	hf_i
f_3, f_4, f_5 : Average, minimal, maximal time distance between hint requests for task i	$\frac{\sum_x hf_i^{(x)}}{hs_i}, \min_x (hf_i^{(x)}), \max_x (hf_i^{(x)})$
f_6 : Number of wrong inputs for task i	$wrong_i$
f_7 : Number of wrong result checks of all checks for task i	$\frac{wf_i}{r_i}$
f_8 : Time until first result check for task i	rf_i
f_9, f_{10} : Average, maximal time distance between actions	$\frac{\sum_a action_i^{(a)}}{na_i}, \max_a action_i^{(a)}$

Table 6 Best p-value, R^2 and Adjusted R^2 value for the log-file features mapped to perceived task-difficulty labels per task (with $p < 0.1$, $R^2 \geq 0.12$)

Log-file feature per task	p-value	R^2	Adjusted R^2
Number of hints seen	2.2e-16	0.2531	0.2499
Time until first hint request	1.2e-12	0.1962	0.1927
Maximal time distance between actions	7.5e-09	0.1344	0.1307

the examples of different classes in the space of the example vectors. By means of the kernel trick and a kernel function also non-linear problems can be solved by an SVM. Originally, an SVM solves binary classification problems but it can be extended to multi-class classification problems (see Chang and Lin (2011)). For our experiments we used the library LIBSVM and applied SVMs with an RBF-kernel and for each SVM used we conducted a grid search (according to Hsu et al. (2011)) on each fold to estimate the optimal values for the hyper parameters. Just as SVMs, DCTs and KNN can be used for classification. DCTs are tree structures in which inner nodes represent tests on an attribute, each branch represents the outcome of the test, each leaf node represents a class label and a path from the root to a leaf represents a classification rule. In KNN an example is classified by a majority vote of its neighbours, that is the example is assigned to the class most common among its k nearest neighbours. These k nearest neighbours are the k closest training examples in the feature space. Also for DCT and KNN we conducted a grid search on each fold for the hyper parameters. We used for our experiments the implementations from Weka (Hall et al. (2009)) for DCT and KNN. For each experiment we applied a leave-one-student-out cross-validation and oversampling for balancing the data (see the following sections for more details). *Oversampling* (see e.g. Barandela et al. (2004)) is needed if the training set is unbalanced, i.e. if there are many examples of one class but less examples of another class, as for the data used in this work. In those cases the class with less examples has to be oversampled to balance the training set. The results of our experiments are reported in terms of classification test error (number of incorrectly classified examples of all

Table 7 p-value, R^2 and Adjusted R^2 for the best combination of log-file features per task

#	Log-file features per task	p-value	R^2	Adjusted R^2
7	Time spent on the task, number of hints seen, time until first hint request, average time distance between hint requests, minimal time distance between hint requests, time until first result check, average time distance between actions	2.2E-16	0.4059	0.3875

examples within the test data) and F-measure. The *F-measure* is the harmonic mean of *Recall* and *Precision*:

$$Recall = \frac{true\ positives}{true\ positives + false\ negatives}, \quad Precision = \frac{true\ positives}{true\ positives + false\ positives} \quad (2)$$

$$F\text{-measure} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (3)$$

In formula (2) *true positives* are examples of a class *c* which are correctly classified as belonging to class *c*, *false negatives* are examples of a class *c* incorrectly classified as not belonging to class *c* and *false positives* are examples of another class than *c* which are incorrectly classified as belonging to class *c*. Hence, *Recall* reports how many of all examples of a class *c* (*true positives* + *false negatives*) are correctly recognised as members of that class. *Precision* shows how many of the examples classified as belonging to class *c* (*true positives* + *false positives*) actually belong to class *c*. These first experiments delivered an average classification test error of 25.24 % for the SVM, of 29.62 % for KNN and of 30.62 % for the DCT as well as F-measures for the 3 classes of 0.67, 0.68, 0.84 for the SVM, of 0.66, 0.67, 0.80 for KNN and of 0.60, 0.63, 0.83 for the DCT (see Table 9). These results show, that the features are suitable for recognising the perceived task-difficulty per task and that SVM performed best. However, for the application in a real system the errors still are not satisfactory. Hence, in the next section we present an approach for improving the perceived task-difficulty recognition performance.

Improving Perceived Task-Difficulty Recognition Performance

For improving the multi-class perceived task-difficulty recognition performance of SVM, DCT and KNN we generalised the SVM plait approach of Janning et al. (2014e). The generalised structure is shown in Fig. 4. For the generalised classification plait the classifiers (*Clasf* in Fig. 4) – which can be SVM, DCT or KNN – are interweaved within a plait structure by combining the classification decisions of classifiers in previous plait layers with the feature vectors and feeding these combined new feature vectors into further classifiers. In this way the classification performance is improved over the plait layers, as classifiers in subsequent layers learn how to consider the classification decisions of previous classifiers to improve their own classification performance. The feature vectors for the classifiers $Clasf_1^{(0)}$, $Clasf_2^{(0)}$, $Clasf_3^{(0)}$ in the first plait layer P_0 stem only from the original feature vector $\mathbf{x}_i = (x_i^0, \dots, x_i^l)$, $(l + 1) =$ number of features. The original feature vector \mathbf{x}_i is divided into as many vectors as there are classifiers in one plait layer, i.e. the first input vectors are:

$$x^{(1)} = \mathbf{x}_i^{Clasf_1^{(0)}} = (x_i^0, \dots, x_i^{(1 \cdot \frac{l}{s})}), \quad (4)$$

$$x^{(2)} = \mathbf{x}_i^{Clasf_2^{(0)}} = (x_i^{(1 \cdot \frac{l}{s})+1}, \dots, x_i^{(2 \cdot \frac{l}{s})}), \quad (5)$$

$$x^{(s)} = \mathbf{x}_i^{Clasf_s^{(0)}} = (x_i^{((s-1) \cdot \frac{l}{s})+1}, \dots, x_i^l). \quad (6)$$

If l is too small then the input vectors $x^{(1)}, x^{(2)}, \dots, x^{(s)}$ also may overlap to ensure that there are enough feature values within one vector for a good classification performance of the single classifiers. The input feature vectors for the subsequent layers within the plait structure are different. That is, the input feature vectors $\mathbf{x}_i^{\text{Class}_1^{(d)}}, \mathbf{x}_i^{\text{Class}_2^{(d)}}, \dots, \mathbf{x}_i^{\text{Class}_s^{(d)}}$ for the classifiers $\text{Class}_1^{(d)}, \text{Class}_2^{(d)}, \dots, \text{Class}_s^{(d)}$ in plait layer P_d are enhanced by $(s - 1)$ additional inputs:

$$\mathbf{x}_i^{\text{Class}_1^{(d)}} = (\mathbf{x}_i^{\text{Class}_1^{(0)}}, \hat{y}_i^{\text{Class}_2^{(d-1)}}, \dots, \hat{y}_i^{\text{Class}_s^{(d-1)}}) = (x_i^0, \dots, x_i^{(1 \cdot \frac{1}{s})}, \hat{y}_i^{\text{Class}_2^{(d-1)}}, \dots, \hat{y}_i^{\text{Class}_s^{(d-1)}}) \tag{7}$$

$$\begin{aligned} \mathbf{x}_i^{\text{Class}_2^{(d)}} &= (\mathbf{x}_i^{\text{Class}_2^{(0)}}, \hat{y}_i^{\text{Class}_1^{(d-1)}}, \hat{y}_i^{\text{Class}_3^{(d-1)}}, \dots, \hat{y}_i^{\text{Class}_s^{(d-1)}}) \\ &= (x_i^{(1 \cdot \frac{1}{s})+1}, \dots, x_i^{(2 \cdot \frac{1}{s})}, \hat{y}_i^{\text{Class}_1^{(d-1)}}, \hat{y}_i^{\text{Class}_3^{(d-1)}}, \dots, \hat{y}_i^{\text{Class}_s^{(d-1)}}) \end{aligned} \tag{8}$$

$$\mathbf{x}_i^{\text{Class}_s^{(d)}} = (\mathbf{x}_i^{\text{Class}_s^{(0)}}, \hat{y}_i^{\text{Class}_1^{(d-1)}}, \dots, \hat{y}_i^{\text{Class}_{(s-1)}^{(d-1)}}) = (x_i^{((s-1) \cdot \frac{1}{s})+1}, \dots, x_i^l, \hat{y}_i^{\text{Class}_1^{(d-1)}}, \dots, \hat{y}_i^{\text{Class}_{(s-1)}^{(d-1)}}) \tag{9}$$

These additional inputs $\hat{y}_i^{\text{Class}_1^{(d-1)}}, \hat{y}_i^{\text{Class}_2^{(d-1)}}, \dots, \hat{y}_i^{\text{Class}_s^{(d-1)}}$ are the outputs, i.e the predicted class labels, of the classifiers $\text{Class}_1^{(d-1)}, \text{Class}_2^{(d-1)}, \dots, \text{Class}_s^{(d-1)}$ of the previous plait layer $P_{(d-1)}$. This means that $\text{Class}_1^{(d)}, \text{Class}_2^{(d)}, \dots, \text{Class}_s^{(d)}$ take into account the classification decisions – wrong or correct – of the previous classifiers $\text{Class}_1^{(d-1)}, \text{Class}_2^{(d-1)}, \dots, \text{Class}_s^{(d-1)}$ to improve their own classification. The generalised classification plait possesses 3 hyper-parameters: (1) the number $(q + 1)$ of plait layers, (2) the number s of classifiers within one plait layer, and (3) the number n of features as input for one classifier within the plait.

For a better understanding of the plait we will explain it by an example. Lets assume the number of plait layers is 7 ($q + 1 = 7$), the number of classifiers within one plait layer is 3 ($s = 3$) and the number of features as input for one classifier is 4 ($n = 4$), like in the experiments below. There are 11 log-file features for perceived task-difficulty recognition per task (Table 5): f_0, \dots, f_{10} . As $s = 3$ and $n = 4$, (f_0, \dots, f_{10}) is split into 3 input vectors with 4 features each: $(f_0, \dots, f_3), (f_3, \dots, f_6), (f_7, \dots, f_{10})$ ($x^{(1)}, x^{(2)}, x^{(3)}$ in Fig. 4). When an example has to be classified, its values for the 11 features are fed by means of these 3 input vectors into the 3 different classifiers ($\text{Class}_1^{(0)}, \text{Class}_2^{(0)}, \text{Class}_3^{(0)}$ in Fig. 4) in the first plait layer P_0 . The outputs y_1, y_2, y_3 of these classifiers, i.e. their classification decisions (1 for *hard*, 2 for *appropriate* or 3 for *easy*), are added to the original input vectors, delivering 3 new input vectors for the classifiers $\text{Class}_1^{(1)}, \text{Class}_2^{(1)}, \text{Class}_3^{(1)}$ in the second plait layer P_1 : $(f_0, \dots, f_3, y_2, y_3), (f_3, \dots, f_6, y_1, y_3), (f_7, \dots, f_{10}, y_1, y_2)$. In this way these classifiers take into account the classification decision of the classifiers in the previous layer to improve their own classification decision. If y'_1, y'_2, y'_3 are the improved outputs of $\text{Class}_1^{(1)}, \text{Class}_2^{(1)}, \text{Class}_3^{(1)}$, the classifiers $\text{Class}_1^{(2)}, \text{Class}_2^{(2)}, \text{Class}_3^{(2)}$ in the next plait layer P_2 are fed with the 3 input vectors $(f_0, \dots, f_3, y'_2, y'_3), (f_3, \dots, f_6, y'_1, y'_3), (f_7, \dots, f_{10}, y'_1, y'_2)$. This approach is repeated until the seventh layer. Subsequently, the outputs y''_1, y''_2, y''_3 of the classifiers in the seventh layer

are concatenated to a new vector (y''_1, y''_2, y''_3) which is fed into a second stage classifier delivering the final classification decision y . The above described plait approach will be proven by experiments in the following section.

Multi-Class Classification with SVM, DCT and KNN plait

In a second step we conducted experiments for solving the multi-class classification problem by using the presented general plait structure with different classifiers (SVM, DCT and KNN) to show that the plait structure also works for multi-class classification problems and with different classifiers. The data set for our experiments consists of 234 examples, 20 with label *hard*, 93 with label *appropriate* and 121 with label *easy* (as mentioned above, we consider only examples of tasks which were not skipped, i.e. examples with label *too hard* and *too easy* are not considered). The numbers per student are shown in Table 8. According to Janning et al. (2014e) we used for the experiments 3 classifiers within one plait layer as well as 4 features as input for one classifier within the plait. We applied the plait in each case with 20 plait layers and observed that the classification test error converges to one certain error value (after at most 7 layers). As in the first experiments we conducted a leave-one-student-out cross-validation and we applied oversampling for balancing the data (see Table 8). We compared SVM, KNN and DCT plait against applying one single SVM, KNN or DCT (applied to the full input feature vector) as well as the two ensemble methods majority vote, which takes the majority decision of several single classifiers (3, according to the number of SVMs within one plait layer) as classification output, and stacking, which learns to combine the classification decisions of several single classifiers (also 3) by a further classifier which gets the outputs of the other classifiers as input (like $Clasf_0$ in Fig. 4). The inputs for the 3 single classifiers of the ensemble methods are the same as for the single classifiers within the first plait layer. The results of the experiments are shown in Table 9, Figs. 2 and 3. In Table 9 the average classification test errors and F-measures of a single SVM, DCT and KNN, of majority and stacking ensembles and of SVM, DCT and KNN plait are reported. As

Table 8 Numbers of train (original and after oversampling) and test examples per student and class (1: *hard*, 2: *appropriate*, 3: *easy*) for the experiments

Student	1	2	3	4
Accumulated label	appropri./easy	appropriate	hard/appropr.	easy
Test (class 1, 2, 3)	1, 9, 30	3, 20, 10	10, 14, 7	0, 2, 31
Train (class 1, 2, 3) original	19, 84, 91	17, 73, 111	10, 79, 114	20, 91, 90
Train (class 1, 2, 3) oversampling	91, 91, 91	111, 111, 111	114, 144, 144	91, 91, 91
Student	5	6	7	AVG
Accumulated label	hard	appropri./easy	appropri./easy	
Test (class 1, 2, 3)	6, 19, 0	0, 23, 14	0, 6, 29	3, 13, 17
Train (class 1, 2, 3) original	14, 74, 121	20, 70, 107	20, 87, 92	17, 80, 104
Train (class 1, 2, 3) oversampling	121, 121, 121	107, 107, 107	92, 92, 92	104, 104, 104

Table 9 Average classification test errors and F-measures of a single SVM, DCT and KNN, of majority and stacking ensembles and of SVM, DCT and KNN plait (best results in bold)

	SVM	KNN	DCT
Single classifier classification test error (%)	25.24	29.62	30.62
Single classifier F-measures (class 1, 2, 3)	0.67, 0.68, 0.84	0.66, 0.67, 0.80	0.60, 0.63, 0.83
Majority ensemble classification test error (%)	24.47	30.85	33.53
Majority ensemble F-measures (class 1, 2, 3)	0.73, 0.69, 0.85	0.67, 0.66, 0.81	0.59, 0.59, 0.77
Stacking ensemble classification test error (%)	20.41	22.67	28.93
Stacking ensemble F-measures (class 1, 2, 3)	0.71, 0.76, 0.87	0.64, 0.73, 0.85	0.61, 0.64, 0.77
Plait classification test error (%)	14.54	12.84	20.57
Plait F-measures (class 1, 2, 3)	0.89 , 0.84, 0.90	0.83, 0.86, 0.95	0.67, 0.75, 0.83
Class. test error improvement compared to single classifier	42 %	57 %	33 %
F-measure improvement compared to single classifier	33 % , 24 %, 7 %	26 %, 28 % , 17 %	12 %, 19 %, 0 %

one can see, SVM performed best as single classifier. SVM, DCT and KNN stacking outperform the single classifiers as well as majority ensemble, whereas SVM, DCT and KNN plait outperform all other classification models and improve the classification performance substantially. The best performing plait on average is the KNN plait, except for class *hard* where the SVM plait performed best. Overall, both SVM and KNN plait performed similarly good whereas DCT plait showed a weaker performance. Figure 2 displays the classification test errors per student of single SVM, KNN, DCT, of SVM, KNN, DCT majority and stacking and of SVM, KNN, DCT plait. One can see in Fig. 2 that in each case the plait outperformed the other models, except for SVM and DCT applied to the data of student 7. In Fig. 3 one can see the F-measures per student of single SVM, KNN, DCT, of SVM, KNN, DCT majority and stacking and of SVM, KNN, DCT plait for each class. Also this figure shows that in most cases the plait performed best and builds an upper bound. Only in the case of DCT the plait seems not always to work well. However, on average also the

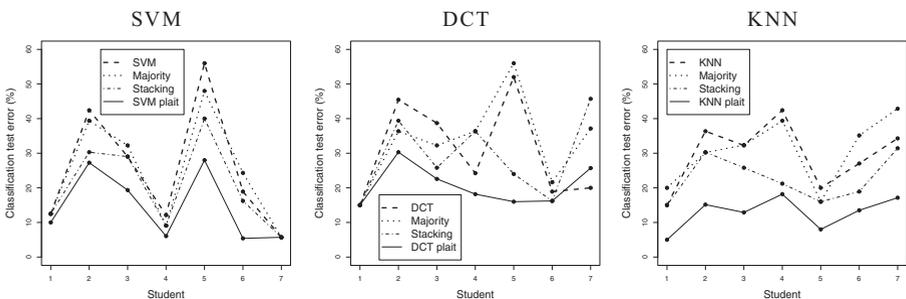


Fig. 2 Classification test errors per student of single SVM, KNN, DCT, of SVM, KNN, DCT majority and stacking ensemble and of SVM, KNN, DCT plait

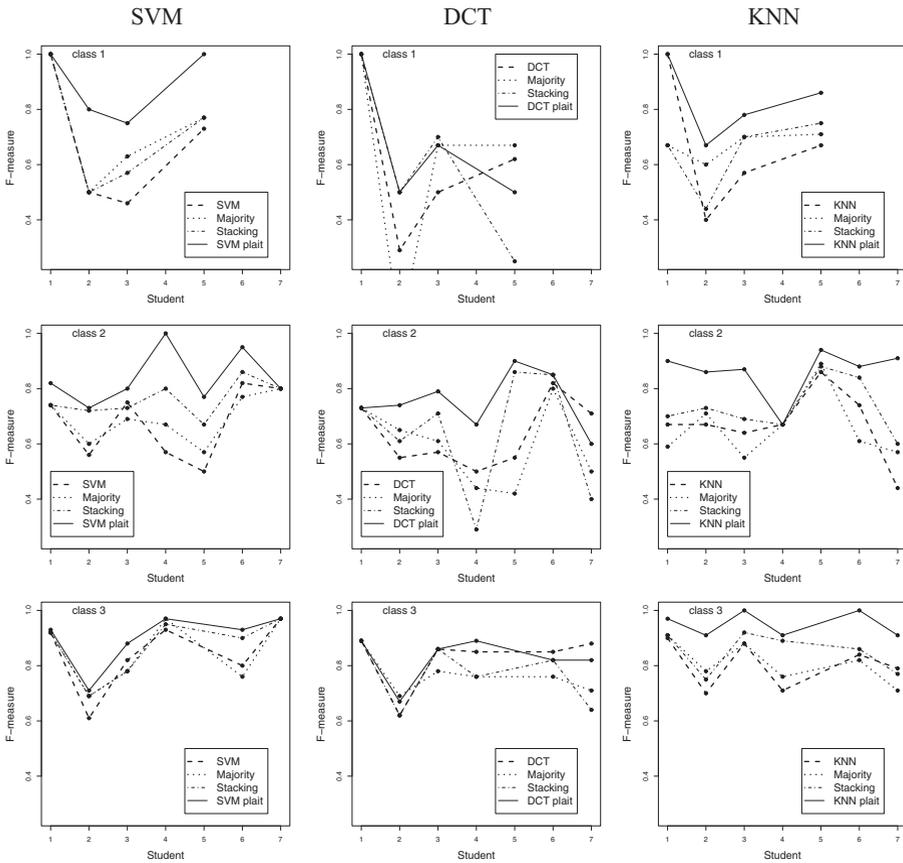


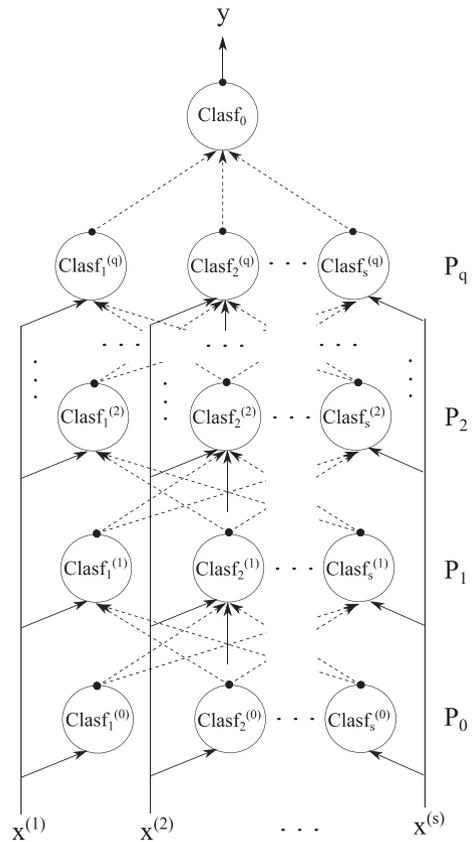
Fig. 3 F-measures per student of single SVM, KNN, DCT, of SVM, KNN, DCT majority and stacking ensemble and of SVM, KNN, DCT plait for each class

DCT plait can improve the F-measures (see Table 9). In summary, the results of the experiments show that the plait approach (Fig. 4) is able to improve the classification performance substantially (in terms of classification test error as well as F-measures).

Future Work in Supporting Task Sequencing by Perceived Task-Difficulty Recognition

As mentioned above, the proposed perceived task-difficulty recognition shall support the task sequencing approach for adaptive intelligent tutoring systems described in Schatten and Schmidt-Thieme (2014) (see also Schatten et al. (2014), Schatten et al. (2015)). This approach applies a performance prediction for future tasks and decides by means of the predictions and the *Vygotsky Policy* (based on Vygotsky’s *Zone of Proximal Development*) which task is chosen next. The performance prediction in Schatten and Schmidt-Thieme (2014) is done by matrix factorization which is trained

Fig. 4 Architecture of the generalised classification plait. The plait is composed of $q + 1$ layers P_0, P_1, \dots, P_q (q is a hyper parameter). Each layer contains s classifiers (Clasf), which get different feature vectors of length n as input (also s and n are hyper parameters). In every plait layer from P_1 on the classifiers are retrained with enhanced input feature vectors. The enhancement is information from the previous layer, namely the outputs (the predicted class labels) of the classifiers in the previous plait layer. After the last plait layer P_q a further classifier (Clasf₀) is attached to achieve one common output y delivering the final classification result



with former performances of all students and learns how to predict the performances of a student for future tasks by means of the performance information about other students and the performance history of the considered student. That is, the output are predicted performances for each task not yet seen by the student. In a next step the approach applies the *Vygotsky Policy*: it compares each of these performance predictions with a threshold which stands for the performance of a student in the *Zone of Proximal Development*, and chooses that task as next task which is closest to this threshold. The threshold is a constant obtained from the outputs of a sensitiveness analysis with simulated students (Schatten and Schmidt-Thieme (2014)) and discussions with teachers and intelligent tutoring system experts (Schatten et al. (2015)). The idea behind this approach is to keep students within the *Zone of Proximal Development*, and therefore those tasks are chosen from that we predict that the student will achieve a performance showing that the student is within the *Zone of Proximal Development*. We aim to support this approach by perceived task-difficulty recognition in three different ways:

- providing a personalised and adaptive threshold for the *Vygotsky Policy*,
- correcting the *Vygotsky Policy* threshold value,
- detecting misconceptions.

Personalised Adaptive Threshold The threshold for the *Vygotsky Policy* in Schatten and Schmidt-Thieme (2014) is the same constant value for every student. However, the performance value related to the *Zone of Proximal Development* may be different for different students and even different for different tasks for the same student. Hence, using a personalised and adaptive threshold for each student would supposedly be better. By means of our proposed perceived task-difficulty recognition we can provide such a personalised adaptive threshold. The first step is to identify students *similar* to the considered student, where *similar* means with similar performances for the tasks the considered student has already worked on. For these *similar students* a perceived task-difficulty recognition has to be done for all the tasks which they worked on. The value for the threshold representing the *Zone of Proximal Development* for the considered student can then be computed by taking the average of the performance scores which the k most *similar students* reached when they felt *appropriately challenged* (task was *appropriate*), as we assume that a student who does not feel *over-challenged* (class *hard*) nor *under-challenged* (class *easy*), but *appropriately challenged* is in the *Zone of Proximal Development*. The *similar students* identification and the computation of the personalised adaptive threshold value for the *Vygotsky Policy* can be done by means of methods like k -Nearest-Neighbour. Additionally, the average of the performance values which the considered student reached when he/she felt *appropriately challenged* in former tasks can be incorporated by including it into the average computation.

Correcting the Vygotsky Policy Threshold Value If the perceived task-difficulty recognition classifies the student as *over-challenged*, although the *Vygotsky Policy* sequencer has chosen a task with a predicted performance related to the *Zone of Proximal Development*, then the threshold value may have been too small. So the threshold value should be corrected to a higher value, i.e. the student gets next a task with a larger predicted performance which is assumed to be easier for the student. Conversely, if the perceived task-difficulty recognition classifies the student as *under-challenged*, although the *Vygotsky Policy* sequencer has chosen a task with a predicted performance related to the *Zone of Proximal Development*, then the threshold value may have been too large. In this case the threshold value should be corrected to a lower value, i.e. the student gets next a task with a smaller predicted performance which is assumed to be more difficult for the student.

Detecting Misconceptions The comparison between (a) the performance of the student, and (b) the recognised perceived task-difficulty can indicate situations in which the student may not fully understand the underlying principles of the task to work on although he/she thinks so. This could be the case, when the student felt to be *under-* or *appropriately challenged* but performed very badly. In those cases the system should provide hints or assign the student a next task that rather helps the student to understand the underlying concepts. In the opposite case instead, when the student felt *over-challenged* but performed very well, the system should provide positive feedback to the student.

Conclusions

We presented an approach for recognising the perceived task-difficulty of students interacting with an intelligent tutoring system. For this purpose we proposed features easily extracted from the log-file. These features report about the actions of the students like *average and maximal time distance between actions*, where the underlying idea is that the longer the time distance between actions of the student the more he/she probably thinks about the task and the more challenging is that task for the student. By means of statistical tests and different classification experiments we showed that these features are able to describe the perceived task-difficulty of a student. The statistical tests showed that there are statistically significant feature combinations and the classification experiments delivered good classification performances. Nevertheless, additional tests and experiments with larger data should be conducted to make the observations of this work generalisable. Additionally, we presented a kind of ensemble method for the different tested classification models and showed that this ensemble method is able to further improve the classification performance and therefore the perceived task-difficulty recognition substantially. Finally, we presented concepts for supporting task sequencing in adaptive intelligent tutoring systems by the proposed perceived task-difficulty recognition. Future steps will be to integrate the presented perceived task-difficulty recognition into an intelligent tutoring system and evaluate the proposed approaches for supporting task sequencing by the output of the perceived task-difficulty recognition.

Acknowledgments The research leading to the results reported here has received funding from the European Union Seventh Framework Programme (FP7/2007 – 2013) under grant agreement no. 318051 – iTalk2Learn project (www.italk2learn.eu).

References

- Ai, H., Litman, D.J., Forbes-Riley, K., Rotaru, M., Tetreault, J., & Purandare, A. (2006). Using System and User Performance Features to Improve Emotion Detection in Spoken Tutoring Dialogs. *INTERSPEECH 2006*, pp. 797–800.
- Arroyo, I., Woolf, B.P., Burelson, W., Muldner, K., Rai, D., & Tai, M. (2014). *A Multimedia Adaptive Tutoring System for Mathematics that Addresses Cognition, Metacognition and Affect*. *Int. J. Artif. Intell. Educ.* (24), (pp. 387–426): Springer.
- Baker, R.S.J.D., Gowda, S., Wixon, M., Kalka, J., Wagner, A., Salvi, A., Aleven, V., Kusbit, G., Ocumpaugh, J., & Rossi, L. (2012). Towards Sensor-Free Affect Detection in Cognitive Tutor Algebra. *Proceedings of the 5th International Conference on Educational Data Mining (EDM 2012)*, pp. 126–133.
- Barandela, R., Valdovinos, R., Sánchez, J., & Ferri, F. (2004). The imbalanced training sample problem: Under or over sampling? *Structural, Syntactic, and Statistical Pattern Recognition*, Springer, Lecture Notes in Computer Science, pp. 806–814.
- Boser, B.E., Guyon, I., & Vapnik, V. (1992). *A training algorithm for optimal margin classifiers*. *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, (pp. 144–152): ACM Press.
- Breiman, L., Friedman, J.H., Olshen, R.A., & Stone, C.J. (1984). *Classification and regression trees*. Wadsworth International Group, Belmont CA, ISBN 0-412-04841-8.
- Chang, C.C., & Lin, C.J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3), 1–27.

- Cichocki, A., Zdunek, R., Phan, A.H., & Amari, S.I. (2009). *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*: Wiley.
- Cooper, D.G., Muldner, K., Arroyo, I., Woolf, B.P., & Burleson, W. (2010). *Ranking feature sets for emotion models used in classroom based intelligent tutoring systems*. *User Modeling, Adaptation, and Personalization (UMAP 2010)*, (pp. 135–146): Springer.
- Cortes, C., & Vapnik, V. (1995). Support-vector network. *Machine Learning*, 20, 273–297.
- Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1), 21–27. doi:[10.1109/TIT.1967.1053964](https://doi.org/10.1109/TIT.1967.1053964).
- D’Mello, S., Picard, R., & Graesser, A. (2007). Towards An Affect-Sensitive AutoTutor. *IEEE Intelligent Systems*, 22(4), 53–61.
- D’Mello, S.K., Craig, S.D., Witherspoon, A., McDaniel, B., & Graesser, A. (2008). Automatic detection of learner’s affect from conversational cues. *User Model User-Adap Inter*. doi:[10.1007/s11257-007-9037-6](https://doi.org/10.1007/s11257-007-9037-6).
- D’Mello, S.K., & Graesser, A. (2012). Language and Discourse Are Powerful Signals of Student Emotions during Tutoring. *IEEE Transactions on Learning Technologies, IEEE Computer Society*, 5(4), 304–317.
- Epp, C., Lippold, M., & Mandryk, R.L. (2011). Identifying Emotional States Using Keystroke Dynamics. *Proceedings of the 2011 Annual Conference on Human Factors in Computing Systems (CHI 2011)*, pp. 715–724.
- Forbes-Riley, K., & Litman, D.J. (2004). Predicting emotion in spoken dialogue from multiple knowledge sources. In *Proceedings of HLTNAACL 2004* (pp. 201–208).
- Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I.H. (2009). The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11(1).
- Hsu, C.W., Chang, C.C., Lin, C.J., & A Practical Guide to Support Vector Classification. Technical report, Department of Computer Science, National Taiwan University. (<http://www.csie.ntu.edu.tw/~cjlin/>) (2011).
- Hu, X., Tang, L., Tang, J., & Liu, H. (2013). Exploiting Social Relations for Sentiment Analysis in Microblogging. *Proceedings of the Sixth ACM WSDM Conference (WSDM ’13)*.
- Janning, R., Schatten, C., & Schmidt-Thieme, L. (2013). HNNP – A Hybrid Neural Network Plait for Improving Image Classification with Additional Side Information. *Proceedings of the IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2013)*, Washington DC, USA, pp. 24–29.
- Janning, R., Schatten, C., & Schmidt-Thieme, L. (2014a). *Automatic Subclasses Estimation for a Better Classification with HNNP Proceedings of the 21th International Symposium on Methodologies for Intelligent Systems (ISMIS 2014), Lecture Notes in Artificial Intelligence*: Springer.
- Janning, R., Schatten, C., & Schmidt-Thieme, L. (2014b). Multimodal Affect Recognition for Adaptive Intelligent Tutoring Systems. *Extended Proceedings of the 7th International Conference on Educational Data Mining (EDM 2014)*, pp. 171–178.
- Janning, R., Schatten, C., & Schmidt-Thieme, L. (2014c). Feature Analysis for Affect Recognition Supporting Task Sequencing in Adaptive Intelligent Tutoring Systems. *Proceedings of the European Conference on Technology Enhanced Learning (EC-TEL 2014)*, pp. 179–192.
- Janning, R., Schatten, C., & Schmidt-Thieme, L. (2014d). *Local Feature Extractors Accelerating HNNP for Phoneme Recognition Proceedings of the 37th German Conference on Artificial Intelligence (KI 2014), Lecture Notes in Artificial Intelligence*: Springer.
- Janning, R., Schatten, C., Schmidt-Thieme, L., Backfried, G., & Pfannerer, N. (2014e). An SVM Plait for Improving Affect Recognition in Intelligent Tutoring Systems. *Proceedings of the IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2014)*.
- Janning, R., Schatten, C., & Schmidt-Thieme, L. (2015). Recognising perceived task difficulty from speech and pause histograms. *Proceedings of the 17th International Conference on Artificial Intelligence in Education (AIED 2015)*.
- Litman, D.J., & Forbes-Riley, K. (2004). Predicting Student Emotions in Computer-Human Tutoring Dialogues. *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL’04)*, pp. 351–358.
- Luz, S. (2013). Automatic identification of experts and performance prediction in the multimodal math data corpus through analysis of speech interaction. *Second International Workshop on Multimodal Learning Analytics*, Sydney Australia.

- Mavrikis, M. (2008). Data-driven modelling of students' interactions in an ILE. Proceedings of the International Conference on Educational Data Mining (EDM 2008), pp. 87–96.
- Moore, J.D., Tian, L., & Lai, C. (2014). Word-level Emotion Recognition Using High-Level Features. Computational Linguistics and Intelligent Text Processing (CICLing 2014), pp. 17–31.
- Morency, L.P., Oviatt, S., Scherer, S., & Weibel, N. (2013). ICMI 2013 grand challenge workshop on multimodal learning analytics. Proceedings of the 15th ACM on International conference on multimodal interaction (ICMI 2013), pp. 373–378.
- Pardos, Z.A., Baker, R.S.J., San Pedro, M., Gowda, S.M., & Gowda, S.M. (2014). Affective States and State Tests: Investigating How Affect and Engagement during the School Year Predict End-of-Year Learning Outcomes. *Journal of Learning Analytics*, 1(1), 107–128. Inaugural issue.
- Purandare, A., & Litman, D. (2006). Humor: Prosody Analysis and Automatic Recognition for F * R * I * E * N * D * S *. Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006), pp. 208–215.
- Qi, F., Bao, C., & Liu, Y.A. (2004). novel two-step SVM classifier for voiced/unvoiced/silence classification of speech. International Symposium on Chinese Spoken Language Processing, pp. 77–80.
- Robison, J., McQuiggan, S., & Lester, J. (2009). Evaluating the consequences of affective feedback in intelligent tutoring systems. 3rd International Conference on Affective Computing and Intelligent Interaction and Workshops (ACII 2009), pp. 1–6.
- Sadegh, M., Ibrahim, R., & Othman, Z.A. (2012). Opinion Mining and Sentiment Analysis: A Survey. *International Journal of Computers & Technology*, 2(3).
- Saif, H., He, Y., & Alani, H. (2012). Semantic Sentiment Analysis of Twitter. Proceedings of the 11th International Semantic Web Conference (ISWC 2012).
- San Pedro, M.O.C., Baker, R.S.J.D., Bowers, A., & Heffernan, N. (2013). Predicting College Enrollment from Student Interaction with an Intelligent Tutoring System in Middle School. Proceedings of the 6th International Conference on Educational Data Mining (EDM 2013), pp. 177–184.
- Schatten, C., & Schmidt-Thieme, L. (2014). Adaptive Content Sequencing without Domain Information. Proceedings of the Conference on computer supported education (CSEDU 2014).
- Schatten, C., Janning, R., Mavrikis, M., & Schmidt-Thieme, L. (2014). Matrix Factorization Feasibility for Sequencing and Adaptive Support in Intelligent Tutoring Systems. Proceedings of the 7th International Conference on Educational Data Mining (EDM 2014), pp. 385–386.
- Schatten, C., Janning, R., & Schmidt-Thieme, L. (2015). Integration and Evaluation of a Matrix Factorization Sequencer in Large Commercial ITS. Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI15).
- Schuller, B., Batliner, A., Steidl, S., & Seppi, D. (2011). *Recognising realistic emotions and affect in speech: State of the art and lessons learnt from the first challenge*. *Speech Communication*: Elsevier.
- Ting, K.M., & Witten, I.H. (1999). Issues in stacked generalization. *Journal of artificial intelligence research*, 10, 271–289.
- Vygotsky, L.I.S. (1978). *Mind in society: The development of higher psychological processes*: Harvard university press.
- Wang, Y., & Heffernan, N. (2011). Extending Knowledge Tracing to allow Partial Credit: Using Continuous versus Binary Nodes. *Artificial Intelligence in Education, Lecture Notes in Computer Science*, 7926, 181–188.
- Wolf, B., Burleson, W., Arroyo, I., Dragon, T., Cooper, D., & Picard, R. (2009). Affect-aware tutors: recognising and responding to student affect. *International Journal of Learning Technology*, 4(3/4), 129–164.
- Worsley, M., & Blikstein, P. (2011). What's an Expert? Using Learning Analytics to Identify Emergent Markers of Expertise through Automated Speech, Sentiment and Sketch Analysis. Proceedings of the 4th International Conference on Educational Data Mining (EDM '11), pp. 235–240.