Vygotsky based Sequencing without Domain Information: A Matrix Factorization Approach

Carlotta Schatten, Ruth Janning, and Lars Schmidt-Thieme

Information Systems and Machine Learning Lab,
University of Hildesheim, Germany
{schatten, janning, schmidt-thieme}@ismll.uni-hildesheim.de
http://www.ismll.uni-hildesheim.de

Abstract. Sequencing contents, like tasks, hints, and feedbacks, is an open issue for Intelligent Tutoring Systems. The common approach is based on domain analysis by experts, who characterize each content with skills involved and a difficulty level. In addition, Machine Learning based sequencers require a specific dataset collection to create users' models and a sequencing policy, which needs to be tested online with strong ethical requirements and a high number of users. In this paper we design a simulated learning environment with customizable scenarios. We also show that a performance prediction method can be used to crate offline fully personalized students' models and sequence contents without domain engineering/authoring effort. The performance prediction method is enhanced by a score-based policy inspired by Vygotsky's concept of Zone of Proximal Development and shows promising results compared to curriculum based policies in the designed simulated environment.

Keywords: Sequencing, Performance Prediction, Intelligent Tutoring Systems, Matrix Factorization

1 Introduction

Intelligent Tutoring Systems (ITS) are more and more becoming of crucial importance in education. Apart from the possibility to practice any time, adaptivity and individualization are the main reasons for their widespread availability as app, web service and software. The system generally is composed of an internal user model and a sequencer, that, according to the given information, sequences the contents with a policy. On that side many efforts have been put into Bayesian Knowledge Tracing (BKT), starting with not personalized and single skills user modeling. The limit of this problem formulation became clear soon, also because the contents evolved together with the technology. Multiple skills contents were developed, e.g. multiple step exercises and simulated exploration environment for learning. In order to maintain the single skill formulation systems fell back on scaffolding, i.e. a built in structure was inserted in order to clearly distinguish within the content between the different steps/skills required. As a consequence, the engineering and authoring effort to develop an ITS increased exponentially obliging a meticulous analysis of the contents in order to subdivide and design them in

clearly separable skills.
Other efforts have been put into adaptive sequencing. Its main approach, based on Reinforcement Learning, can be reconnected to robotics, which has an availability of accurate simulators and tireless test subjects. The same cannot be said for ITS where, generally, apart from adults, also children of any age are involved.
In this paper we propose a novel method of sequencing based on Matrix Factorization Performance Prediction and Vygotsky's concept of Zone of Proximal Development. The main contributions are:

1. A content sequencer based on a performance prediction systems that (1) can be set up and preliminary evaluated in a laboratory, (2) models multiple skills and individualization without engineering/authoring effort, (3) adapts to each combination of contents, levels and skills available.
2. Simulated environment with multiple skill contents and students' knowledge representation, where knowledge and performance are modeled in a continuous way.
3. Experiments on different scenarios with direct comparison with informed baseline.

The paper is structured as follows: in Section 2 one can find a brief state of the art description, in Section 3 the explanation of the sequencer problem, in Section 4 the simulated learning process, in Section 5 the performance based policy and predictor, in Section 6 the experimental results and least the conclusions.

2 Related Work

Many Machine Learning techniques have been used to ameliorate ITS, especially in order to extend learning potential for students and reduce engineering efforts for designing the ITS. The most used technology for sequencing is Reinforcement Learning (RL), which computes the best sequence trying to maximize a previously defined reward function. Both model–free and model–based [12,1] RL were tested for content sequencing. Unfortunately, the model–based RL necessitates of a special kind of data sets called exploratory corpus. Available data sets are log files of ITS which have a fixed sequencing policy that teachers designed to grant learning. They explore a small part of the state–action space and yield to biased or limited information. For instance, since a novice student will never see an exercise of expert level, it is impossible to retrieve the probability of a novice student solving some contents. Without these probabilities the RL model cannot be built [2]. Model–free RL, instead, assumes a high availability of students on which one can perform an on-line training. The model does not require an exploratory corpus but needs to be built while the users are playing with the designed system. Given the high cost of an experiment with humans, most authors exploit simulated single skill students based on different technologies like Artificial Neural Networks or self developed student models [16,12]. Particularly similar to our approach is [12], where contents are sequenced with a particular model–free RL based on the actor critic algorithm [9], which was selected because of its faster convergence in comparison with the classic Q–Learning algorithm [20]. Unfortunately, RL algorithms still need many episodes to converge and will always need preliminary trainings on simulated students.

Our developed content sequencer is based on student performance predictions. An example of state of the art method is Bayesian Knowledge Tracing (BKT) and its extensions. The algorithm is built on a given prior knowledge of the students and a data set of binary student performances. It is assumed that there is a hidden state representing the knowledge of a student and an observed state given by the recorded performances. The model learned is composed by slip, guess, learning and not learning probability, which are then used to compute the predicted performances [4]. In the BKT extensions also difficulty, multiple skill levels and personalization are taken into account separately [24,13,14,5]. BKT researchers have discussed the problem of sequencing both in single and in multiple skill environment in [8]. In a single skill environment the most not mastered skill is selected, whereas in the multiple skill this behavior would present a too difficult content sequence. Consequently, the contents with a small number of not mastered skills are selected. Moreover, [8] points out how in ITS multiple skill exercises are modeled as single skill ones in order to overcome BKT limitations. We would like to stress that the sequencing requires an internal skills representation and consequently, together with the performance prediction algorithm, is domain dependent.
Another domain dependent algorithm used for performance prediction is the Performance Factors Analysis (PFM). In the latter the probability of learning is computed using the previous number of failures and successes, i.e. the representation of score is binary like in BKT [15]. Moreover, similarly to BKT, a table connecting contents and skills is required.
Matrix Factorization (MF) is the algorithm used in this paper for performance prediction. It has many applications like, for instance, dimensionality reduction, clustering and also classification [3]. The most common use is for Recommender Systems [10] and recently this concept was extended to ITS [21]. We selected this algorithm for several reasons:

1. Domain independence: ability to model each skill, i.e. no engineering or authoring effort in individuating the skills involved in the contents.
2. Having comparable results with BKT latest implementations [22].
3. Possibility to build the system with a common data set, i.e. without an exploratory corpus.
4. Small computational time on a 3rd Gen Ci5/4GB laptop and Java implementation: 0.43 s for building the model with already 122000 lines, negligible time for performance prediction.

3 Content Sequencing in ITS

The designed system consists of two main blocks. The first one is the environment and is represented by the students playing with the ITS. First step toward a working prototype requires testing in a laboratory. Since optimal control problems can only be evaluated online, i.e. the sequence optimality can be measured only after a student worked with it, we designed a simulated learning process that is described in Sec. 4. We excluded the possibility of collecting an exploratory corpus because making practice with very easy and very difficult exercises in random order could be frustrating for the students, who could be children. After a first validation with real students, only a common data

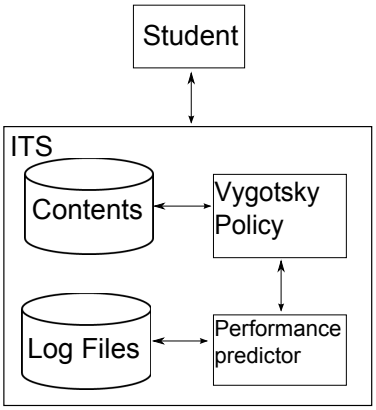

Fig. 1: System structure in a block diagram.

set collection will be necessary to set up the system with new contents, giving also the possibility to calibrate the environment and later use it for new sequencing methods. In this paper we use the word content to refer to the activities a student interacts with, although our main focus here is task sequencing. Taking advantage of the simulated learning process characteristics described, we can later interpret contents as different ITS elements. As explained in [17], a content could be a hint, a feedback, a topic or a task, whose sequencing could in each case take advantage of the designed system as we will discuss later.
The second block consists of different modules, i.e. the available contents, the previous interactions of the students with the system (log files), the student Performance Predictor and the Sequencer Policy. We chose a specific Performance Predictor and policy, but nothing is against using other ones in the future. When a student plays with the system the next exercise is proposed to him by the sequencer according to a policy. The Performance Predictor needs the log files of students playing with the contents considered to predict their scores in the next contents. The policy is applied in an adaptive way thanks to the information on the predicted scores shared between Performance Predictor and Sequencer. In the following Sections we will describe the different blocks represented in Fig. 1.

4 Simulated Learning Process

Given the necessity of preliminary evaluation in a laboratory, it is of crucial importance to have a simulated environment able to model reality with a certain degree of fidelity. For our system we required a score and skill representation between 0 and 1, to be able to test following aspects:

1. Possibility to use score as single success indicator for sequencing
2. Ability to model a multiple skill domain and students' knowledge by the performance predictor
3. Possibility to change number of skills involved to test flexibility
4. Possibility to test also noisy processes

We designed a simulated student based on the following assumptions. (1) A content is either of the correct difficulty for a student, or too easy, or too difficult. (2) A student cannot learn from too easy contents and learns from difficult ones proportionally to his knowledge level. (3) It is impossible to learn from a content more than the required skills to solve it. (4) The total knowledge at the beginning is different than zero. (5) The general ability on connected skills helps solving and learning from a content. The last assumption is more plausible because we assume to sequence activities of the same domain. For instance, in order to solve a fraction addition, a student needs more related skills: multiplication, fraction expansion etc. It is unlikely for a student to do a fraction expansion without knowing how multiplication works. At the same time the knowledge of multiplication will help him solving the steps on fraction expansion.
A student simulator is a tuple (S,C,y,τ) where, given a set $S \subseteq [0,1]^K$ of students, s_i is a specific student described as a vector φ^t. The latter is of dimension K, where K is the number of skills involved. $C \subseteq [0,1]^K$ is a set of contents, where c_j is the j–th content, defined with a vector ψ_j of K elements representing the skills required. $\varphi_{i,k} = 0$ means student's i skill level k is zero, whereas $\varphi_{i,k} = 1$ means having full ability. $\tau : \mathbb{S} \times \mathbb{C} \to \mathbb{S}$ is a function defining the follow-up state $\varphi^{t+1} = \varphi^t + \tau$ of a student $s_i \in \mathbb{S}$ after working on contents c_j^t. In particular $\mathbb{S}$ and $\mathbb{C}$ are the spaces of the students and contents respectively. Finally, a function y defines the performance $y(\varphi_i, \psi_j)$. y and τ can be formalized as follows:

$$\begin{aligned} y(\varphi_i, \psi_j) &:= \max(1 - \frac{||\alpha||}{||\varphi_i||}, 0) \\ \tau(\varphi_i, \psi_j)_k &:= y(\varphi_{ik}, \psi_{jk})\alpha_k \\ \tilde{y} &:= y\varepsilon \end{aligned} \tag{1}$$

where

$$\alpha_k^{i,j} = \max(\psi_{jk} - \varphi_{ik}, 0) \tag{2}$$

and ε is proportional to the beta distribution $\mathcal{B}(p,q)$. We selected p and q in order to have $\tilde{y} \sim \mathcal{B}\left(y, \sigma^2\right)$, where σ^2 is the variance, i.e. the amount of noise. We chose the beta distribution because it is defined between zero and one as the score. Consequently it will not change the codomain of the y function. The characteristic of the formulas are the following. (1) The performance of a student on a content decreases proportionally to his skill deficiencies w.r.t. the required skills. (2) The student will improve all the required skills of a content proportionally to his performance and his skill-specific deficiency up to the skill level a content requires. (3) As a consequence it is not possible to learn from a content more than the difference from the required and possessed skills. (4) A further property of this model is that contents requiring twice the skills level that a student has, i.e. $\left\|\psi_j\right\| \geq 2\left\|\varphi_i\right\|$, are beyond the reach of a student. For this reason his performance will be zero ($y = 0$). With a simple experiment without noise, we can show

the plausibility of the designed simulator. We inserted values in Eqs. 1 as follows. Let us consider a system with two skills and represent the student knowledge as $\varphi = \{0.3, 0.5\}$. As it is possible to see in Tab. 1 with the increase of the content difficulty the learning

c_j	d_c	y	τ_k
$[0.1, 0.1]$	0.2	1	$[0, 0]$
$[0.5, 0.6]$	1.1	0.617	$[0.12, 0.0617]$
$[0.5, 0.7]$	1.2	0.515	$[0.1, 0.1]$
$[0.9, 0.9]$	1.8	0	$[0, 0]$

Table 1: Simulated learning process with two skills. A simulated student with $\varphi = \{0.3, 0.5\}$ scores y and learning τ after interacting with different contents c_j.

increases and the score decreases until $\|\psi_i\| \geq 2 \left\|\varphi_j\right\|$. The maximal difficulty level is equal to the number of skills since a single skill value cannot be greater than one.

5 Vygotsky Policy and Matrix Factorization

5.1 Sequencer

The designed sequencer is defined as follows. Let $C \subseteq \mathbb{C}$ and $S \subseteq \mathbb{S}$ be respectively a set of contents and students defined in Section 4, d_{c_j} be the difficulty of a content defined as $d_{c_j} = \sum_{k=0}^{K} \psi_{j,k}$, $\tilde{y} : \mathbb{S} \times \mathbb{C} \to [0, 1]$ be the performance or the score of a student working on the content, and T be the number of time steps assuming that the student is seeing one content every time step. The content sequencing problem consists in finding a policy:

$$\pi^* : (\mathbb{C} \times [0, 1]) \to \mathbb{C}. \tag{3}$$

that maximize the learning of a student within a given time T without any environment knowledge, i.e. without knowing the difficulties of the contents and the required skills to solve them. A common problem in designing a policy for ITS is retrieving the knowledge of the student from the given information, e.g. score, time needed, previous exercises, etc. The previous mentioned data types are just an indirect representation of the knowledge, which cannot be automatically measured, but needs to be modeled inside the system. Hence, integrating the curriculum and skills structure is the cause of the high costs in designing the sequencer. In this paper we try to keep the contents in the Vygotskys Zone of Proximal Development (ZPD) [23], i.e. the area where the contents neither bore or overwhelm the learner. We mathematically formalized the concept with the following policy, that we called Vygotsky Policy (VP):

$$c^{t*} = \text{argmin}_c \left| y_{th} - \hat{y}^t(c) \right| \tag{4}$$

where y_{th} is the threshold score, i.e. the score that keeps the contents in the ZPD. The policy will select at each time step the content with the predicted score $\hat{y}^t$ at time t most similar to y_{th}. We will discuss further in the experiment session how to tune this hyper

parameter and its meaning.
The peculiarity of the VP is the absence of the difficulty concept. Defining the difficulty for a content in a simulated environment as ours is easy, because we mathematically define the skills required. In the real case it is not trivial and quite subjective. Also the required skills are considered as given in the other state of the art methods like PFM and BKT, where a table represents the connection between contents and skills required. Without skills information not only BKT and PFM performance prediction cannot be used in our formalization, also sequencing methods [8] have no information to work with.

5.2 Matrix Factorization as Performance Predictor

Matrix Factorization (MF) is a state-of-the-art method for recommender systems. It predicts which is the future user ratings on a specific items based on his previous ratings and the previous ratings of other users. The concept has been extended to student performance prediction, where a student next performance, or score is predicted. The matrix $Y \in \mathbb{R}^{n_s \times n_c}$ can be seen as a table of n_c total contents and n_s students used to train the system, where for some contents and students performance measures are given. MF decomposes the matrix Y in two other ones $\Psi \in \mathbb{R}^{n_c \times P}$ and $\Phi \in \mathbb{R}^{n_s \times P}$, so that $Y \approx \hat{Y} = \Psi\Phi$. Ψ and Φ are matrices of latent features. Their elements are learned with gradient descend from the given performances. This allows computing the missing elements of Y and consequently predicting the student performances (Fig. 2). The optimization function is represented by:

$$\min_{\Psi_j, \Phi_i} \sum_{j \in \mathbb{C}} (y_{ij} - \hat{y}_{ij})^2 + \lambda(\|\Psi\|^2 + \|\Phi\|^2) \tag{5}$$

where one wants to minimize the regularized squared error on the set of known scores. The prediction function is represented by:

$$\hat{y}_{ij} = \mu + \mu_{cj} + \mu_{si} + \sum_{p=0}^{P} \varphi_{ip}^T \Psi_{jp} \tag{6}$$

where μ, μ_c and μ_s are respectively the average performance of all contents of all students, the learned average performance of a content, and learned average performance of a student. The two last mentioned parameters are also learned with the gradient descend algorithm.
The MF problem does not deal with time, i.e. all the training performances are considered equally. In order to keep the model up to date, it is necessary to re-train the model at each time step. MF has a personalized prediction, i.e. a small number of exercises needs to be shown to each student in order to avoid the so called cold–start problem. Although some solutions to these problems have been proposed in [21,11], we will show in the experiment session that these aspects do not affect the performance of the system, neither they reduce its applicability. From now on we will call the sequencer utilizing the VP policy and the MF performance predictor VPS, i.e. Vygotsky Policy based Sequencer.

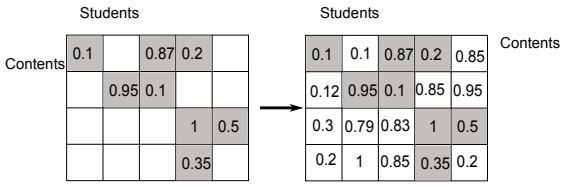

Fig. 2: Table of scores given for each student on contents (left), completed table by the MF algorithm with predicted scores (right).

6 Experiment Session

In this section we show how the single elements work in detail. We start with the student simulator, continue with the VP and end with some experiments with performance prediction in different scenarios and noise. A scenario is represented by a number of contents n_c, a number of difficulty levels n_d, a number of skills n_k, and a number of students for each group n_t[1]. All the first experiments will have no noise, i.e. $\tilde{y} = y$.

6.1 Experiments on the Simulated Learning Process

To prove the operating principle of the simulator we tested basic sequencing methods in a particular scenario. The one we chose is described in Fig. 3, with $n_d = 7$ and $n_c = 150$. For representation purposes we created the contents with increasing difficulty, so that IDs implicitly indicates the difficulty[2]. The scenario mimics an interesting situation for sequencing, i.e. when more apparently equivalent exercises are available. The two policies we used are (1) Random (RND), where contents are selected randomly, and (2) the in range policy (RANGE), where each second content is selected in difficulty order. This strategy is informed on the domain because it knows the difficulty of the contents. We initialized the students and contents skills with an uniform random distribution between 0 and 1. Again for representation purposes we show the average total knowledge of the students that is represented by average of the students skills sum at each time step. We chose to perform the tests on 10 skills, i.e. the maximal total knowledge possible is equal to 10. We considered the scenario mastered when the total knowledge of the student group is greater than or equal to the 95% of the maximal total knowledge.
Fig. 4 shows the total knowledge of two groups of $n_t = 200$ students, one group was trained with random policy the other one with the in range policy. RANGE is characterized by a low variance in the learning process. RND, instead, has a high variance because the knowledge level of the students at each time step is given by chance. It is shown that the order in which the student practices on the contents is important for the total final learning. Fig. 4 also shows how the practice on too many contents of the same difficulty level, after a while, saturates the knowledge acquisition. All these aspects demonstrate that the learning progress is plausibly simulated.

[1] The MF was previously trained with n_s students that were used to learn the characteristic of the contents. Consequently, the dimensions of the MF during the simulated learning process are: $\Psi \in \mathbb{R}^{n_c \times P}$ and $\Phi \in \mathbb{R}^{(n_s+n_t)\times P}$, so that $Y \approx \hat{Y} = \Psi\Phi$.

[2] A content with ID 2 is easier than a content with ID 100, see Fig. 3

6.2 Sensitivity Analysis on the Vygotsky Policy

In order to evaluate the VP we created two more sequencing methods that exploit information not available in reality. The best sequencing knows exactly which is the content maximizing the learning for a student, for this reason we called it Ground Truth (GT). Vygotsky Policy Sequencer Ground Truth (VPSGT), instead, uses the Vygotsky Policy and the true score y of a student to select the following content. GT and VPSGT can be considered the upper bound of the sequencer potential in a scenario. In order to select the correct value of y_{th} we plot the average knowledge level at time $t = 11$ for the policy with different y_{th}. From Fig. 5 one can see that the policy is working for $y_{th} \in [0.4, 0.7]$, this because of the relationship between Eqs. 1 of the student simulator. In a real environment the interpretation of these results is twofold. First we assume y_{th} will be approximately the score keeping the students in the ZDP. Second, from a RL perspective, this value would allow finding the trade–off between exploring new concepts and exploiting the already possessed knowledge. Moreover, as one can see in Fig. 6, the policy obtains good results if compared with GT for some y_{th}, but for others the policy is outside the ZPD and the students do not reach the total knowledge of the scenario. In some experiments we noticed that the width of the curve in Fig. 5 decreased so that the outer limits of the y_{th} interval create a sequence outside the ZPD. As consequence we selected the value $y_{th} = 0.5$ that was successful in most of the scenarios.

6.3 Vygotsky Policy based Sequencer

The scenario we selected for the tests with the VPS has $n_c = 150$, $n_d = 6$, $n_k = 10$ and $n_t = 400$. In order to train the MF–model a training and test data set need to be created. We used $n_s = 300$ students who learned with all the contents in order of difficulty. We used 66% of the data to train the MF–model and the remaining 34% to evaluate the Root Mean Squared Error (RMSE) for selecting the regularization factor λ and the learning rate of the gradient descent algorithm. We performed a full Grid Search and selected the parameters shown in Tab. 2. The sequencing experiments are done on a separate group of n_t students. In order to avoid the cold start problem 5 contents are shown to them and their scores added to the training set of the MF. For $T = 40$ the best content c_j^{*t} is selected with the policy VP for the n_t students, using the predicted performance $\hat{y}_{ij}^t$. In order to avoid the deterioration of the model, after each time step the model is trained again once all students saw an exercise. A detailed description of the algorithm of the sequencer can be found in Alg. 1, where Y_0 is the initial data set.
As one can see in Fig. 7 the VPS selects the first content similarly to RANGE. Then the prediction allows to skip unnecessary contents speeding up the learning. Once the total knowledge arrives around 95%, the selection policy cannot find contents that fit to the requirements. Consequently the students learn as slow as the RND group, as one can see from the saturating curve. In Fig. 8 GT selects the contents in difficulty order skipping the unnecesary ones. The average sequence of the VPS, instead, is also with approximately increasing difficulty but in an irregular way. This is due to the error in the prediction performance. In conclusion the proposed sequencer gains 63% over RANGE and 150% over RND. The presented experiments show how the MF is able, without

Parameters	Choice
Learning Rate	0.01
Latent Features	60
Regularization	0.02
Number of Iteration	10

Table 2: Parameters MF

Algorithm 1: Vygotsky Policy based Sequencer

Input: $\mathbb{C}$, Y_0 π, s_i, T
Train the MF using Y_0;
for *t = 1 to T* **do**
  for *All* $c \in \mathbb{C}$ **do**
    Predict $\hat{y}\left(c_j, s_i\right)$ Eq. 6;
  end
  Find c^{t*} according to Eq. 5;
  Show c^{t*} to s_i with Eq. 1;
  Add $y\left(s_i, c^{t*}\right)$ to Y_t;
  Retrain the MF; // Corrects over- or underestimation by the MF
end

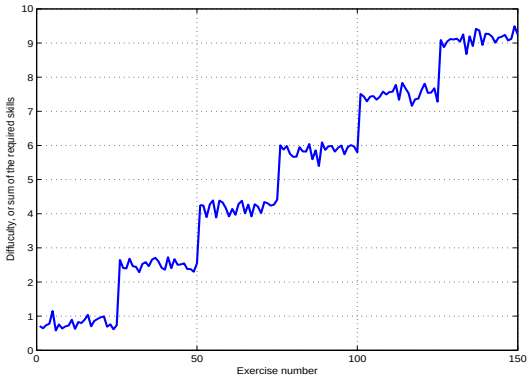

Fig. 3: Scenario: content number and difficulty level.

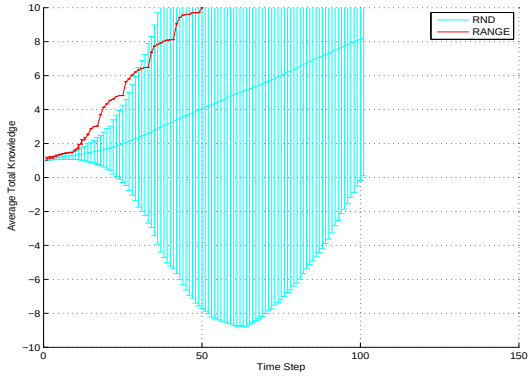

Fig. 4: Comparison between RANGE and RND. Average skills sum, i.e. knowledge, over all the students with variance

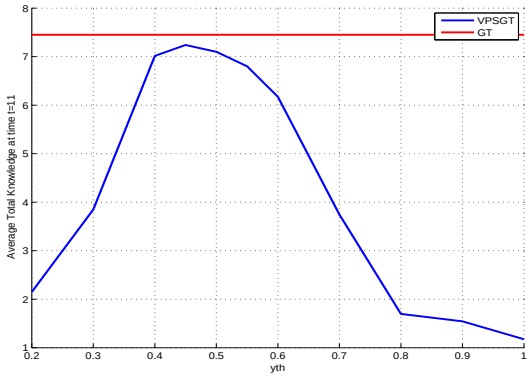

Fig. 5: Policy selection, i.e. the performance of the Vygotsky policy with different y_{th} at the same time step. Different groups of students learned with the Vygotsky policy with y_{th} values going from 0.1 to 0.9. As shown in the figure the knowledge levels change according to the y_{th} selected.

Policy	**Description**
Random (RND)	Contents are selected randomly
In Range (RANGE)	Each second content is selected in difficulty order.
Ground Truth (GT)	Selects the contents according to which is the one maximizing the learning.
Vygotsky Policy based Sequencer Ground Truth (VPSGT)	Chooses the next content using the policy and the real score of a student.
Vygotski Policy based Sequencer (VPS)	Chooses the next content using the policy and the predicted score of a student.

Table 3: Sequencers Description

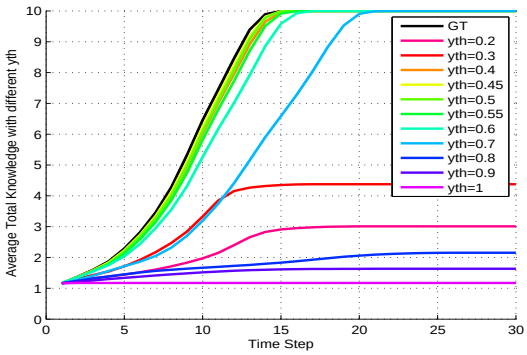

Fig. 6: Effects of the different y_{th} on the final knowledge of the students. The learning curves of the student groups that learned with the different Vygotsky policies.

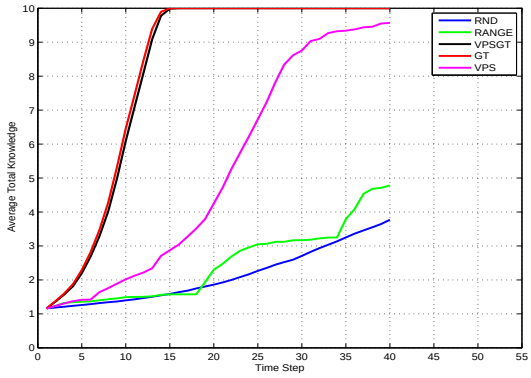

Fig. 7: Average Total Knowledge. How the average learning curve of the students changes over time.

Fig. 8: Average sequence selected by the GT and the VPS. The VPS approximate the optimal sequence that GT computes thanks to the real skills of the students.

domain information, to model the different skills of students and contents and partially mimics the best sequence, which is the one selected by GT in Fig. 8.

6.4 Advanced Experiments

In this section we want to show the correct working of the sequencer changing the parameters of the scenario n_k and n_c and later adding noise. In order to do so we consider the percentage of gain of VPS with respect to RANGE considering a specific time step $t = 30$ with $n_k = 10$ and $n_d = 6$. As one can see in Fig. 10 the gain obtained by the sequencer depends on the available number of contents. Since in RANGE each second content is selected, with $n_c < 60$ there are not enough contents for all time steps. Our sequencer can adapt without problems to the situation. The optimal point for the in range policy is when $n_c = 60$ because there is exactly the necessary number of contents for the student to learn. When $n_c > 60$ the students see many unnecessary contents and consequently learn slower. Fig. 9 with $n_c = 60$, $t = 30$ and $n_d = 6$ shows the dependencies between skills and gain. The experiments demonstrated a high adaptability of the sequencer to the different scenarios.
Last we experimented the results robustness adding noise, i.e. $\tilde{y} = y\varepsilon$. We experimented with $\sigma^2 \in [0, 0.5]$. As one can see in Fig. 11 with $\sigma^2 = 0.1$ the Vygotsky sequencers are still able to produce a correct learning sequence but more time is required. The VPSGT is the one that suffered the most from the introduction of noise, probably related to the selection of y_{th}.

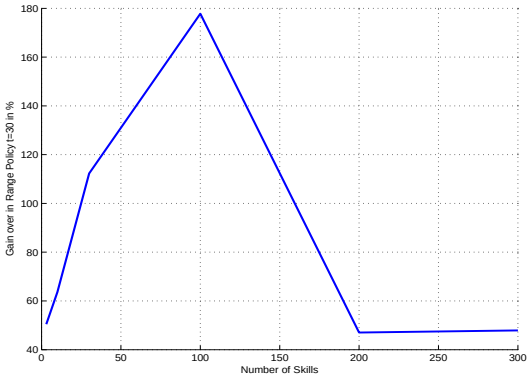

Fig. 9: Gain over RANGE policy varying n_k. The gain is measured at a specific time step in percentage, considering the average knowledge level of the two groups of students, one practicing with the RANGE sequencer and one with the VPS.

6.5 Outlook

VPS has an advantage in comparison to other state of the art methods because it does not require a detailed analysis of the skills involved. Nevertheless, some steps are required for the VPS to be integrated within a learning platform. This aspect has been addressed

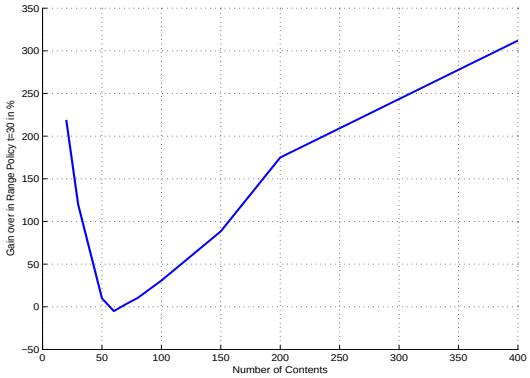

Fig. 10: Gain over RANGE policy varying n_c. The gain is measured at a specific time step in percentage, considering the average knowledge of the two groups of students, one practicing with the RANGE sequencer and one with the VPS.

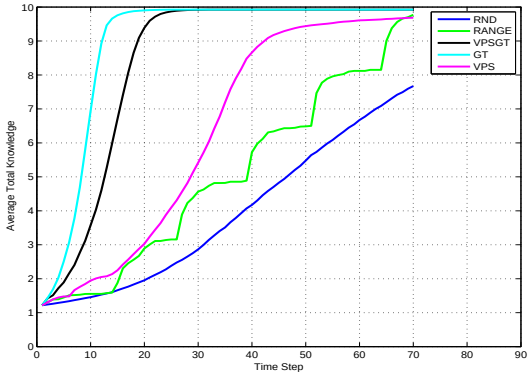

Fig. 11: Effect of noise in the simulated learning process. Beta distribution noise with $\sigma^2 = 0.1$.

in [17] for a commercial ITS, where a first offline feasibility discussion was done. Thanks to VPS domain independence, conceptual integration required minor changes. For technical integration we utilized the work in [19] where a novel minimal invasive integration for a Machine Learning-powered sequencer was presented. Another open question is how to select y_{th}. In [17], we explained how this value should be tailored to the available contents exploiting the passing score set by expert. In [6,7] is proposed to personalize the threshold value by means of the output of an affect recognition applied to students speech input. The proposed rule based policy suggests to increase yth if the student felt under-challenged and decrease it if he felt over-challenged. Alternatively a combination of MF and emotion recognition could be designed.

7 Conclusions

In this paper we presented VPS, a sequencer based on performance prediction and Vygotsky's concept of ZPD for multiple skills contents with continuous knowledge and performance representation. We showed that MF is able dealing with the most actual problems of Intelligent Tutoring Systems, like time and personalization, retrieving automatically skills required and difficulty. We proposed VP, a performance based policy that does not require direct input of domain information, and a student simulator that helps in preliminary off-line evaluation. The designed system achieved time gain over random and in range policy in almost each scenario and is robust to noise. This demonstrates how the sequencer could solve many engineering/authoring efforts. Nevertheless, an experiment with real students is required to better confirm the validity of the assumptions of the simulated learning process. A different evaluation is required for the performance prediction based sequencer. Some work was done in this direction and was mentioned in the previous section. In conclusion, to use VPS, no content analysis is required, since the MF will reconstruct the domain information, thanks to continuous score representation. This will allow the integration of the sequencer in ITS whose content analysis is not affordable. With the results obtained in this paper we plan to extend such an approach also to other intervention strategies to further reduce the engineering efforts in ITS after an complete evaluation on real students.

8 Acknowledgment

This research has been co-funded by the Seventh Framework Programme of the European Commission, through project iTalk2Learn (#318051). www.iTalk2Learn.eu. This paper is an extended version of [18] presented at the 6th International Conference on Computer Supported Education.

References

1. Joseph Beck, Beverly Park Woolf, and Carole R Beal. Advisor: A machine learning architecture for intelligent tutor construction. *AAAI/IAAI*, 2000:552–557, 2000.
2. M. Chi, K. VanLehn, D. Litman, and P. Jordan. Empirically evaluating the application of reinforcement learning to the induction of effective and adaptive pedagogical strategies. *UMAI*.

3. Andrzej Cichocki, Rafal Zdunek, Anh Huy Phan, and Shun-ichi Amari. *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. Wiley. com, 2009.
4. Albert Corbett and John Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *UMAI*, 1994.
5. Ryan SJ D Baker, Albert T Corbett, and Vincent Aleven. More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. In *ITS*, pages 406–415. Springer, 2008.
6. Ruth Janning, Carlotta Schatten, and Schmidt-Thieme Lars. Feature analysis for affect recognition supporting task sequencing. In *ECTEL*, 2014.
7. Ruth Janning, Carlotta Schatten, and Schmidt-Thieme Lars. Multimodal affect recognition for adaptive intelligent tutoring systems. In *FFMI EDM*, 2014.
8. K. Koedinger, P. Pavlik, J. Stamper, T. Nixon, and S. Ritter. Avoiding problem selection thrashing with conjunctive knowledge tracing. In *EDM*, 2011.
9. Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12:1008–1014, 2000.
10. Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.
11. Artus Krohn-Grimberghe, Andre Busche, Alexandros Nanopoulos, and Lars Schmidt-Thieme. Active learning for technology enhanced learning. In *Towards Ubiquitous Learning*, pages 512–518. Springer, 2011.
12. Ankit Malpani, Balaraman Ravindran, and Hema Murthy. Personalized intelligent tutoring system using reinforcement learning. In *FLAIRS*, 2011.
13. Zachary A Pardos and Neil T Heffernan. Modeling individualization in a bayesian networks implementation of knowledge tracing. In *UMAP*. Springer, 2010.
14. Zachary A Pardos and Neil T Heffernan. Kt-idem: introducing item difficulty to the knowledge tracing model. In *UMAP*, pages 243–254. Springer, 2011.
15. Cen H. Pavlik, P. and K. Koedinger. Performance factors analysis-a new alternative to knowledge tracing. In *AIED*, 2009.
16. BH Sreenivasa Sarma and Balaraman Ravindran. Intelligent tutoring systems using reinforcement learning to teach autistic students. In *Home Informatics and Telematics: ICT for The Next Billion*, pages 65–78. Springer, 2007.
17. Carlotta Schatten, Manolis Mavrikis, Ruth Janning, and Lars Schmidt-Thieme. Matrix factorization feasibility for sequencing and adaptive support in its. In *EDM*, 2014.
18. Carlotta Schatten and Lars Schmidt-Thieme. Adaptive content sequencing without domain information. In *CSEDU*, 2014.
19. Carlotta Schatten, Martin Wistuba, Lars Schmidt-Thieme, and Sergio Gutirrez-Santos. Minimal invasive integration of learning analytics services in its. In *ICALT*, 2014.
20. Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. Cambridge Univ Press, 1998.
21. Nguyen Thai-Nghe, Lucas Drumond, Tomas Horvath, Artus Krohn-Grimberghe, Alexandros Nanopoulos, and Lars Schmidt-Thieme. Factorization techniques for predicting student performance. *Educational Recommender Systems and Technologies: Practices and Challenges. IGI Global*, 2011.
22. Nguyen Thai-Nghe, Lucas Drumond, Tomas Horvath, and Lars Schmidt-Thieme. Using factorization machines for student modeling. In *UMAP Workshops*, 2012.
23. L Lev Semenovich Vygotsky. *Mind in society: The development of higher psychological processes*. HUP, 1978.
24. Yutao Wang and Neil T Heffernan. The student skill model. In *ITS2012*, pages 399–404. Springer, 2012.