Comparing Student Models in Different Formalisms by Predicting Their Impact on Help Success

Sébastien Lallé[1,2,3], Jack Mostow[3], Vanda Luengo[1], and Nathalie Guin[2]

[1] LIG METAH, Joseph Fourier University, Grenoble, France
[2] LIRIS, University of Lyon 1, CNRS, Lyon, France
[3] Carnegie Mellon University, Pittsburgh PA, United States of America
{sebastien.lalle,vanda.luengo}@imag.fr, mostow@cs.cmu.edu,
Nathalie.Guin@liris.univ-lyon1.fr

Abstract. We describe a method to evaluate how student models affect ITS decision quality – their raison d'être. Given logs of randomized tutorial decisions and ensuing student performance, we train a classifier to predict tutor decision outcomes (success or failure) based on situation features, such as student and task. We define a decision policy that selects whichever tutor action the trained classifier predicts in the current situation is likeliest to lead to a successful outcome. The ideal but costly way to evaluate such a policy is to implement it in the tutor and collect new data, which may require months of tutor use by hundreds of students. Instead, we use historical data to simulate a policy by extrapolating its effects from the subset of randomized decisions that happened to follow the policy. We then compare policies based on alternative student models by their simulated impact on the success rate of tutorial decisions. We test the method on data logged by Project LISTEN's Reading Tutor, which chooses randomly which type of help to give on a word. We report the cross-validated accuracy of predictions based on four types of student models, and compare the resulting policies' expected success and coverage. The method provides a utility-relevant metric to compare student models expressed in different formalisms.

Keywords: Student models, knowledge tracing, classification, help policy.

1 Introduction

A challenge in the field of Intelligent Tutoring Systems (ITS) is to evaluate student models by their impact on the success of an ITS's decisions – in particular, about which type of help to give students. Individualized help can have a strong impact on learning [1]. The better the tutor adapts its help to the student and situation, the likelier the student will learn from it.

This paper shows how to use logged tutor data and a student model to learn what help to provide in a given situation, and how to compare alternative student models based on the resulting help policies. The paper is organized as follows. Section 2 reviews prior work on learning help policies. Section 3 describes the student models we used in the study. Section 4 discusses the data. Section 5 presents the algorithm for learning a help policy. Section 6 reports results. Section 7 concludes.

K. Yacef et al. (Eds.): AIED 2013, LNAI 7926, pp. 161–170, 2013.

2 Relation to Prior Work

Several papers report positive results from learning individualized help policies.

Andes [7] used a Bayesian network to adapt hints to the student, the problem, and the context, but required human-designed sequences of hint templates; we do not.

ADVISOR [4] and later work [2, 6, 7] used reinforcement learning to adapt a pedagogical agent to optimize student performance metrics such as the time to solve problems. The agent could give hints or to select the next exercise. ADVISOR used only one student model; in contrast, we compare alternative student models. Only Chi *et al.* [6] included features of system behavior, which they found affected feedback success more than task or student features. Barnes and Stamper [2, 7] derived policies from effects of student decisions; in contrast, we learn from tutor decisions.

Project LISTEN's Reading Tutor [19] chose randomly among different types of help on a word. Heiner et al. [13] compared their success rates based on how often the student read a word acceptably at the next encounter. We use this and other information plus a student model to train a policy, not just compare overall success rates.

Razzaq and Heffernan [22] compared two types of feedback, namely scaffolds and hints, and found that students who got scaffolds learned more than students who got hints with pre and post tests, although the difference was not statistically significant. Like Heiner, they compared rates, but between groups rather than within-subject.

Recommender systems can be used to recommend suitable learning resources to a given student in an ITS or web-based learning. Verbert et al. [26] predicted the success of recommendations (in terms of student satisfaction) from student activities. In contrast, we predict the success of help (in terms of student performance) from student traits, task features, help type, and a student model of estimated skills.

Table 1. Summary of prior work on help or hint selection, in terms of features and evaluation

Work	Features used to select help or hints	Methodology to validate learned policy
Gertner *et al.* [11]	Problem goal + current problem state + context + student's mastery of skills	Experiments (pre and post tests)
Beck *et al.* [4]	Student model + current problem state	Simulation (check if probability of success increases with the help) and experiments
Heiner *et al.* [13]	Student level + word difficulty	Use historical data (expected increase in success for unseen students)
Barnes, Stamper *et al.* [3, 23]	Student model + current problem state	Experiments (number of solved problems, errors, and number of hints given with the generated policy vs. default policy)
Chi *et al.* [6]	Student features + domain features + system behavior features	Experiments (pre and post test)
This paper	Student features + domain features + system behavior features	Use historical data (expected increase in success for unseen students)

Table 1 summarizes all this work in terms of the features used in the help or hint policy, and how it was evaluated using on-line experiments or off-line simulation.

Prior research has explored various ways to compare student models [17]. Several papers [5, 12, 21, 27] compare different knowledge tracing models based on goodness of fit. That work frames student modeling as a prediction problem, where the goal is to predict the next observation of student performance (correct or incorrect). Other papers compared the accuracy of models based on constraint-based modeling [14] or Item Response Theory [9]. Results depend on the domain, the datasets, and the model-fitting method. For instance, Pavlik *et al.* found that Performance Factor Models (PFM) beat Bayesian Knowledge Tracing [21], but Gong *et al.* found the opposite, leaving uncertain the reason for this divergence in results [12]. Moreover, we know of no prior quantitative comparisons of different types of student models.

3 Student Models

We now describe the three types of student models we compare in this paper.

Knowledge Tracing [8] is based on a *cognitive model*, which specifies the skills underlying students' successive observable actions. Knowledge tracing uses these observations to update estimated probabilities of the student knowing the skills, based on the *knew* probability of having a skill beforehand, the *learn* probability of acquiring the skill at any given step, the *guess* probability of responding correctly without knowing a skill, and the *slip* probability of responding incorrectly despite knowing it. Knowledge Tracing uses a Bayesian update, while the *Performance Factor Model* (PFM) [21] uses a linear combination of skill difficulty, student proficiency, and past performance (number of previous successes and failures on a given skill).

Constraint-based modeling [20] has no cognitive model of skills underlying steps. Instead, it represents domain constraints whose violation reveals missing knowledge or misconceptions that call for corrective feedback. A constraint-based model represents domain knowledge as a set of constraints (Cr, Cs), where Cr specifies the situations where the constraint is relevant, and Cs specifies the correct answer in those situations. The constraint-based model can infer student knowledge from students' observed actions as the probability of satisfying a constraint when it is relevant.

Finally, the *Control-based Approach* [16] (based on cKc [2]) represents domain knowledge as a set of problems, operators for solving the problems, indicators of how a problem or operator is represented (e.g. as proof vs. diagram in geometry), and skills for deciding whether an answer or action is correct, represented as nodes in a Dynamic Bayesian Network. The Control-based Approach uses observed student actions to update the conditional probability of knowing the skill given the problem, the representation indicators, the operator used, and whether the action is correct.

4 Experimental Data

We use data from Project LISTEN's Reading Tutor [19], which displays text and listens to a child to read it aloud. The Reading Tutor uses automatic speech recognition (ASR) to classify each text word as read correctly or not, and to measure the latency before reading each word. We label a word as *fluent* if the Reading Tutor recognized it as read correctly without help or hesitation. The Reading Tutor can give several types of help on a word, such as say it, give a rhyming hint (e.g., *rhymes with cat*), or sound out its successive phonemes (*/K/ /AE/ /T/*) [13]. Some types of help are infeasible or infelicitous on some words, such as rhyming hints for rhymeless words, syllabifying a one-syllable word, or sounding out a word longer than 4 phonemes. The Reading Tutor chooses randomly among types of help suitable for a given word.

Each such decision generates one randomized controlled trial. To test whether the help helped the child learn the word, we define the outcome of the trial as whether the child read the same word fluently at the next encounter on a later day, thereby excluding recency and scaffolding effects. Thus if a student gets help on word *W* on day *i*, its outcome is the first encounter of word *W* on day *j* where *j*>*i*, as Figure 1 shows. Our data for this paper consist of 30,838 such trials logged by the Reading Tutor in the 2002-2003 school year, from 96 students and 1078 distinct words. To simplify analysis, we omitted trials where a child got help on word *W* more than once on day *i.*

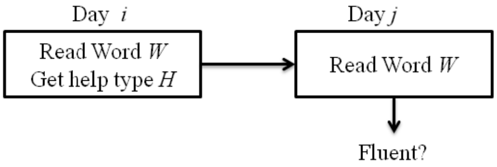

Fig. 1. Help type H on word *W* on day *i* succeeds if *W* is fluent at the first encounter on day *j*

5 Method for Training a Help Policy

We train a help policy as follows. First annotate the logged trials with information from a student model. Then select a set of features (such as student reading level or word length) that affect help success, according to a linear model. Next, use the logged trials, student model, and selected features to learn when help will succeed. Finally, derive a policy from the learned classifier to choose help likeliest to succeed.

5.1 Using Student Models to Annotate Logged Trials

Knowledge diagnosis is the process of inferring or updating a student model from student interactions with an ITS [25]. We considered four knowledge diagnosis techniques: Performance Factor Modeling [21], Bayesian Knowledge Tracing [8], constraint-based modeling [18], and a Control-based model based on cKc [16] (see Table 2). These techniques are generic, so that it is possible to use them on our domain. Diagnosis techniques use various methods to update the student model, such as Bayesian inference or logistic regression. Updating a student model means updating the estimated probability of knowing different skills, based on students' observed performance (such as correct or incorrect answers). These estimates make it possible to predict future performance on those skills.

Table 2. Summary of the four types of student models used in this work

Type of student model	Update method	Output (prediction)	Ref
Performance Factor Model	Linear regression	Probability of answering correctly	[21]
Bayesian Knowledge Tracing	Hidden Markov Model	Probability of answering correctly	[8]
Constraint-based	Constraints	Probability of violating constraints	[18]
Control-based	Dynamic Bayesian Network	Probability of using skills or not, correctly or not	[16]

Constraint-based models are typically updated at the end of exercises. To update them online instead, we associate a power law function with each constraint (knowledge), fit these functions to observed student performance so far, and use them to predict subsequent performance. Another difficulty in our data is that the skills are not directly observable. Our model of oral reading represents a skill as mapping a grapheme to a phoneme. For instance, the word *chemist* maps *ch*→/K/, *e*→/EH/, *m*→/M/, *i*→/IH/, *s*→/S/, and *t*→/T/. However, our speech recognizer only recognizes words. Thus, we used a multiskill approach, meaning that a single observed step (reading a word) may require multiple skills . We estimate each skill independently, predict performance conjunctively (i.e. multiply the estimates of all the skills used in a step), and update each skill separately as if assigning it sole responsibility for the step's success or failure [27].

To fit models that maximize data likelihood, we use EM for Bayesian Knowledge Tracing and Control-based models, and R's stats and igraph packages for Performance Factors Models and Constraint-based models.

5.2 Selecting Features

Help type H on word *W* on day *i* succeeds if *W* is fluent at the first encounter on day *j*. To find which features best predict success, we use stepwise linear regression with success as response variable and features as predictors, and optimize AIC, defined as:

$$\mathrm{AIC} = 2 \times \mathrm{k} - 2 \times \ln(\mathrm{L})$$

Here *k* is the number of parameters of the model and *L* the data likelihood. A one-way ANOVA tests if the features significantly ($p<0.01$) explain success. The initial features were all selected: student's reading level, student proficiency (% of words accepted as fluent when first seen each day), story's difficulty level, word length, word frequency in English, word position in the story, the number of prior encounters of the word, and the word class, defined by which Reading Tutor interventions apply to it.

5.3 Learning Classifiers to Predict Help Success

To predict based on the student model, the selected features, and the type of help whether help will succeed, i.e. lead to reading the word fluently at the next encounter (cf. Figure 1), we trained three types of classifiers – two based on rules (Part [10] and JRip [7]) and one on random trees, using Weka[1]. Here is an example of a learned rule:

[1] `http://www.cs.waikato.ac.nz/ml/weka`

```
1) Word = c145
2) AND Story_Level = B
3) AND Student_Model_Prediction > 0.6
4) AND Help_Type = ″SayWord″
      ⇨ Fluent (22/22)
```

Clause 1 specifies that the rule applies to words in the class "c145," for which the feasible help types (described in [13]) are 1 ("Autophonics"), 4 ("Recue"), and 5 ("RhymesWith"). Clause 2 specifies that the story is at a grade 2 level. Clause 3 specifies that based on prior data, the student model estimates probability over 0.6 that the student will read the word fluently. Clause 4 specifies help type. We compute confidence in a rule as the frequency of success in the training instances to which the rule applies. The rule here predicts with confidence 22/22 that "SayWord" help will succeed. We prune rules with confidence below 0.75 (Weka's default).

5.4 Using a Predictor of Help Success as a Decision Policy for What Help to Pick

The decision policy based on the trained classifier works as follows: Choose the type of help specified by whichever rule applies to the current situation and has the highest confidence according to the training data. If there is more than one such rule, pick randomly among them. An alternative is to train a separate model to predict success for each type of help, and pick a type with the maximum probability of success.

6 Experimental Results

We evaluated our method on Reading Tutor data (cf. section 4). To split the data into two sets, one to train a student model and one to train and test a success predictor, we first sorted the data alphabetically by student initials, and used the first 60% to train a student model. Then we used the remaining 40% to train and test success predictors using 10-fold cross-validation. That is, we partitioned the students into 10 disjoint folds, pooled 9 of them to train a predictor, and tested it on the remaining fold. We repeated this procedure for each fold, and averaged the results. To test how well a student model fit the data, we used it to predict each time the Reading Tutor gave help on a word whether the student read the word fluently at the next encounter of it.

We measure model accuracy as percentage of correct predictions, which Table 3 lists from highest to lowest. We score a probabilistic prediction as correct if it rates the true outcome of the next encounter as likelier than 50%. Varying this probability threshold trades off false positive and false negative errors along an ROC curve. The area under the ROC Curve (AUC) measures the probability that given a fluent and non-fluent instance, the model will correctly identify which is which. AUC of 1 means the model is perfect; AUC of 0.5 means the model is no better than chance.

AIC (defined in section 5.2) measures the goodness of fit to training data based on data likelihood, penalized by the number of parameters k. Here k is the number of model parameters multiplied by the number of skills and the number of students.

Table 3. Predictive accuracy of each student model, and of help success prediction based on it

Type of student model	**Predictiveness of student models**			**Predictors of help success**		
	Accuracy	**AUC**	**AIC**	**Coverage**	**Accuracy**	
Bayesian Knowledge Tracing	84% (± 2.6%)	0.68	5.1 E+4	32%	75% (± 4.1%)	
Control-based model	83% (± 2.9%)	0.67	7.2 E+4	34%	73% (± 4.4%)	} **
Performance Factor model	81% (± 3%)	0.65	5.5 E+4	26%	68% (± 4.4%)	} ***
Constraint-based model	80% (± 2.8%)	0.65	5.6 E+4	25%	65% (± 4.3%)	} ***

Significance on McNemar's test: ** 0.01 < p < 0.05; *** p ≤ 0.01

All 4 diagnostic techniques beat the majority class (76% fluent words in our data). These results are consistent with a previous evaluation of Knowledge Tracing [27] on a different set of Reading Tutor data, which found accuracies ranging from 72% to 87%, but below 35% on non-fluent instances – which might explain why AUC, which measures a model's accuracy in distinguishing positive from negative instances [24], was 0.68 or worse in our data. AIC rated Bayesian Knowledge Tracing highest, penalizing the control-based model because it has more parameters than the other models.

Table 3 evaluates each success predictor by its cross-validated accuracy on help given to held-out students. We show results only from JRip, because it beat the other two classifier methods (by less than 2%). Bayesian Knowledge Tracing did best. Coverage is the proportion of words in the test set to which a rule of a policy applies.

Predictors of help success were less accurate than the student models they used. Evidently, predicting whether a student will read a word fluently at the next encounter is easier than predicting whether help on that word will succeed. A possible reason is data sparseness: we predict success of each help type from the training instances where the Reading Tutor happened to give that type, which may be very few.

To test the statistical reliability of accuracy differences between predictors of help success rate, we used McNemar's test, which checks for significant differences between two classifiers C1 and C2 on the same data using this formula:

$$\chi^2 = (d_1 - d_2)^2 / (d_1 + d_2)$$

Here d_1 is the number of instances classified as positive by C1 but negative by *C2*, and d_2 is the number of instances classified as positive by C2 but negative by C1. The sum $d_1 + d_2$ exceeds 80 in our data, well over the minimum of 10 specified by McNemar [15], so this test can be approximated as a Chi-squared distribution. Each two consecutive predictors in Table 3 differ significantly (p<0.025), assuming negligible effects of statistical dependencies among trials with the same student or word.

Finally, we computed the expected percentage of words read fluently at the next encounter after help based on each learned policy. The difference between expected and actual percentages represents the simulated increase in help success, shown in Table 4. (Simulated means based on historical data rather than on new experiments.) The last row shows results when solely picking types of help with the highest success rate in the training set. We compute the expected help success rate E:

$$E(Fluent \mid h^{*}, S, F)$$

Here S is the student model, F is the set of student and domain features, and $h*$ is the type of help with the highest estimated probability of success in that situation:

$$h^{*} = \mathrm{argmax}_{h}\, E(Fluent \mid h, S, F)$$

Table 4. Expected absolute percentage increase in (simulated) help success

Diagnosis technique (type of student model)	Expected increase in help success	Coverage (% of test set covered by rules)
Bayesian Knowledge Tracing	5.2%	32%
Control-based model	5.1%	34%
Performance Factor Model	4.7%	26%
Constraint-based model	4.5%	25%
Average success in the training set	2.4%	

7 Conclusion

This paper presents new methods to compare student models and induce help policies. Prior work compared the predictive accuracy of student models expressed in the same formalism, e.g. cognitive modeling or Item Response Theory. In contrast, we compare the impact of student models on expected success of tutorial decisions based on them, a measure more directly relevant to utility than predictive accuracy is. We believe quantitative comparison of student models across different formalisms is novel.

We described a method to learn a policy for picking which type of help to give in a given situation, based on types of help, student features, domain features, and a student model, by using this information to learn the probability that help will succeed, and then picking the type of help likeliest to succeed in a given situation. Using data from Project LISTEN's Reading Tutor, we showed that success predictors differ significantly, depending on the student model used. All four learned policies improved the Reading Tutor's expected success compared to its original randomized decisions. A 5.2% increase despite only 32% coverage implies 16.3% increase on the covered test instances; thus better-generalized policies could potentially triple help success.

Our approach has several limitations. It applies only to tutors that decide among multiple types of applicable help. It assumes that the logged decisions were randomized, and that their outcomes can be computed from the ensuing tutorial interactions. The learned policy's coverage and accuracy in predicting whether a given type of help will succeed in a given situation are limited by the number of observations in the logged training data of the tutor giving that type of help in that situation. Thus the method can only learn policies followed sufficiently often in the data to estimate their success. The learned policy is therefore vulnerable to under-covering and overfitting. The accuracy of the cross-validated estimate of the learned policy's expected success is similarly limited by the number of observations of each situation-decision pair in the held-out test data. Both the policy and the estimate of its success assume that the outcomes of the held-out logged instances are representative of future unseen cases. This inductive leap is the price we pay for evaluating the policy based on its simulated rather than actual success. Future work includes trying more accurate student models such as LR-DBN [26], more powerful classifiers such as Support Vector Machines (SVM) or Random Forests, analysis of how student model accuracy affects the accuracy of predicting the success of help, learning more general policies to increase coverage and reduce overfitting, and experiments to test how accurately expected success predicts actual success in practice.

Acknowledgements. This work was supported by the first author's PhD scholarship and travel grant from the Rhône-Alpes Region in France, and by the Institute of Education Sciences, U.S. Department of Education, through Grant R305A080628 to Carnegie Mellon University. The opinions expressed are those of the authors and do not necessarily represent the views of the Institute or U.S. Department of Education. We thank the children, schools, and LISTENers who generated, collected, and organized Reading Tutor data.

References

1. Anderson, J.R., Gluck, K.: What role do cognitive architectures play in intelligent tutoring systems. In: Cognition and Instruction: Twenty-Five Years of Progress, pp. 227–262. Lawrence Erlbaum, Mahwah (2001)
2. Balacheff, N., Gaudin, N.: Students conceptions: an introduction to a formal characterization. Cahier Leibniz 65, 1–21 (2002)
3. Barnes, T., Stamper, J., Lehman, L., Croy, M.: A pilot study on logic proof tutoring using hints generated from historical student data. In: Procs. of the 1st International Conference on Educational Data Mining, Montréal, Canada, pp. 552–557 (2008)
4. Beck, J.E., Woolf, B.P., Beal, C.R.: ADVISOR: a machine-learning architecture for intelligent tutor construction. In: Procs. of the 17th AAAI Conference on Artificial Intelligence, Boston, MA, pp. 552–557 (2000)
5. Cen, H., Koedinger, K., Junker, B.: Comparing two IRT models for conjunctive skills. In: Woolf, B.P., Aïmeur, E., Nkambou, R., Lajoie, S. (eds.) ITS 2008. LNCS, vol. 5091, pp. 796–798. Springer, Heidelberg (2008)
6. Chi, M., VanLehn, K., Litman, D., Jordan, P.: Inducing effective pedagogical strategies using learning context features. In: De Bra, P., Kobsa, A., Chin, D. (eds.) UMAP 2010. LNCS, vol. 6075, pp. 147–158. Springer, Heidelberg (2010)
7. Cohen, W.W.: Fast Effective Rule Induction. In: Procs. of the 12th International Conference on Machine Learning, Tahoe City, CA, pp. 115–123 (1995)
8. Corbett, A.T., Anderson, J.R.: Knowledge tracing: Modeling the acquisition of procedural knowledge. User Modelling and User-Adapted Interaction 4, 253–278 (1995)
9. Desmarais, M.C.: Performance comparison of item-to-item skills models with the IRT single latent trait model. In: Konstan, J.A., Conejo, R., Marzo, J.L., Oliver, N. (eds.) UMAP 2011. LNCS, vol. 6787, pp. 75–86. Springer, Heidelberg (2011)
10. Frank, E., Witten, I.H.: Generating accurate rule sets without global optimization. In: Procs. of the 15th International Conference on Machine Learning, Madison, WI, pp. 144–151 (1998)
11. Gertner, A.S., Conati, C., VanLehn, K.: Procedural help in Andes: Generating hints using a Bayesian network student model. In: Procs. of the 15th National Conference on Artificial Intelligence, Madison, WI, pp. 106–111 (1998)

12. Gong, Y., Beck, J.E., Heffernan, N.T.: How to Construct More Accurate Student Models: Comparing and Optimizing Knowledge Tracing and Performance Factor Analysis. International Journal of Artificial Intelligence in Education 21(1), 27–46 (2011)
13. Heiner, C., Beck, J., Mostow, J.: Improving the help selection policy in a Reading Tutor that listens. In: Procs. of the InSTIL/ICALL 2004 Symposium on NLP and Speech Technologies in Advanced Language Learning Systems, Venice, Italy, pp. 195–198 (2004)
14. Le, N.-T., Pinkwart, N.: Can Soft Computing Techniques Enhance the Error Diagnosis Accuracy for Intelligent Tutors? In: Cerri, S.A., Clancey, W.J., Papadourakis, G., Panourgia, K. (eds.) ITS 2012. LNCS, vol. 7315, pp. 320–329. Springer, Heidelberg (2012)
15. McNemar, Q.: Note on the sampling error of the difference between correlated proportions or percentages. Psychometrika 12(2), 153–157 (1947)
16. Minh Chieu, V., Luengo, V., Vadcard, L., Tonetti, J.: Student modeling in complex domains: Exploiting symbiosis between temporal Bayesian networks and fine-grained didactical analysis. International Journal of Artificial Intelligence in Education 20(3), 269–301 (2010)
17. Mitrovic, A., Koedinger, K., Martin, B.: A comparative analysis of cognitive tutoring and constraint-based modeling. In: Procs. of the 9th International Conference on User Modeling, Johnstown, PA, pp. 313–322 (2003)
18. Mitrovic, A., Ohlsson, S.: Evaluation of a Constraint-Based Tutor for a Database. International Journal of Artificial Intelligence in Education 10(3-4), 238–256 (1999)
19. Mostow, J., Aist, G.: Evaluating tutors that listen: An overview of Project LISTEN. In: Smart Machines in Education: The Coming Revolution in Educational Technology, pp. 169–234. MIT/AAAI Press, Cambridge, MA (2001)
20. Ohlsson, S.: Constraint-based student modeling. NATO ASI Series F Computer and Systems Sciences, vol. 125, pp. 167–189 (1994)
21. Pavlik, P.I., Cen, H., Koedinger, K.: Performance Factors Analysis–A New Alternative to Knowledge Tracing. In: Procs. of the 15th International Conference on Artificial Intelligence in Education, Auckland, New Zealand, pp. 531–538 (2009)
22. Razzaq, L., Heffernan, N.T.: Scaffolding vs. Hints in the Assistment System. In: Ikeda, M., Ashley, K.D., Chan, T.-W. (eds.) ITS 2006. LNCS, vol. 4053, pp. 635–644. Springer, Heidelberg (2006)
23. Stamper, J., Barnes, T., Lehmann, L., Croy, M.: The hint factory: Automatic generation of contextualized help for existing computer aided instruction. In: Procs. of the International 9th Conference on Intelligent Tutoring Systems Young Researchers Track, Montréal, Canada, pp. 71–78 (2008)
24. Swets, J.A.: Measuring the accuracy of diagnostic systems. Science 240(4857), 1285–1293 (1988)
25. VanLehn, K.: Student modeling. In: Foundations of Intelligent Tutoring Systems, pp. 55–78. Lawrence Erlbaum, Mahwah (1988)
26. Verbert, K., Drachsler, H., Manouselis, N., Wolpers, M., Vuorikari, R., Duval, E.: Dataset-driven research for improving recommender systems for learning. In: Procs. of the 1st International Conference on Learning Analytics and Knowledge, Banff, Canada, pp. 44–53 (2011)
27. Xu, Y., Mostow, J.: Comparison of methods to trace multiple subskills: Is LR-DBN best? In: Procs. of the 5th International Conference on Educational Data Mining, Chania, Greece, pp. 41–48 (2012)