Bring It on! Challenges Encountered While Building a Comprehensive Tutoring System Using *ReaderBench*

Marilena Panaite[1], Mihai Dascalu[1,2(✉)], Amy Johnson[3], Renu Balyan[3], Jianmin Dai[3], Danielle S. McNamara[3], and Stefan Trausan-Matu[1,2]

[1] Faculty of Automatic Control and Computers, University "Politehnica" of Bucharest, 313 Splaiul Independenței, 60042 Bucharest, Romania
marilena.panaite@cti.pub.ro, {mihai.dascalu, stefan.trausan}@cs.pub.ro
[2] Academy of Romanian Scientists, Splaiul Independenței 54, 050094 Bucharest, Romania
[3] Institute for the Science of Teaching and Learning, Arizona State University, PO Box 872111, Tempe, AZ 85287, USA
{amjohn43, renu.balyan, jianmin.dai, dsmcnama}@asu.edu

Abstract. Intelligent Tutoring Systems (ITSs) are aimed at promoting acquisition of knowledge and skills by providing relevant and appropriate feedback during students' practice activities. ITSs for literacy instruction commonly assess typed responses using Natural Language Processing (NLP) algorithms. One step in this direction often requires building a scoring mechanism that matches human judgments. This paper describes the challenges encountered while implementing an automated evaluation workflow and adopting solutions for increasing performance of the tutoring system. The algorithm described here comprises multiple stages, including initial pre-processing, a rule-based system for pre-classifying self-explanations, followed by classification using a Support Virtual Machine (SVM) learning algorithm. The SVM model hyper-parameters were optimized using grid search approach with 4,109 different self-explanations scored 0 to 3 (i.e., poor to great). The accuracy achieved for the model was 59% (adjacent accuracy = 97%; Kappa = .43).

Keywords: Natural language processing · Intelligent tutoring systems
Self-explanations · Support vector machines · *ReaderBench*

1 Introduction

This study presents the challenges faced while implementing a comprehensive processing pipeline for automatically scoring self-explanations in the context of Interactive Strategy Training for Active Reading and Thinking (iSTART), a tutoring system that helps students learn and practice using comprehension strategies in the context of self-explanation. The workflow described in this paper integrates advanced Natural Language Processing

C. Penstein Rosé et al. (Eds.): AIED 2018, LNAI 10947, pp. 409–419, 2018.
https://doi.org/10.1007/978-3-319-93843-1_30

(NLP) techniques [1], a wide range of textual complexity indices available in the *ReaderBench* framework [2, 3], fine-tuned heuristics, and Support Vector Machine (SVM) classification [4] applied on primary data consisting of student self-explanations.

In the next section, we discuss the instructions in iSTART along with its practical applications for learning processes. In Sect. 3, the end-to-end development of the feedback system is explained, providing insights into the training dataset, the rule-based system, and the limitations encountered during the process. Section 4 presents the experimental results and analyses use of grid search optimization over the SVM training process. The last section concludes the paper and presents future improvements for the machine-learning model.

2 iSTART

iSTART [5] was designed to improve the quality of self-explanations and the effective use of comprehension strategies while reading a challenging text. iSTART is an intelligent tutoring system that provides extended practice applying high-order comprehension strategies during self-explanation [5, 6]. Because students are provided with automated feedback using natural language processing (NLP), the system affords the ability to provide individualized reading strategy instruction to multiple classrooms of students, each of them interacting in parallel with iSTART [5, 7]. Initial training modules in iSTART provide instruction on five comprehension strategies: comprehension monitoring, paraphrasing, prediction, bridging, and elaboration. These strategies promote and scaffold the generation of inferences, which helps students to construct more complete and accurate mental representations of text [8].

Instruction in iSTART proceeds from training modules to an initial practice phase, and then to extended practice. During the initial training videos, animated agents explain and present examples of the reading strategies that improve comprehension of difficult science texts [5]. Within both initial and extended practice, the student practices generating self-explanation using the repertoire of comprehension strategies. In one of iSTART's generative practice modules, coached practice (see Fig. 1), the animated agent provides feedback that is driven by NLP to evaluate the self-explanations. During these interactions, iSTART's feedback encourages students to use strategies that enhance comprehension.

This paper describes recent advances to improve the accuracy of the NLP algorithm driving assessment and delivery of just-in-time feedback [9, 10]. The major computational challenge faced in the practice modules is providing relevant and appropriate feedback based on the quality of the self-explanation. Assessment of self-explanation quality proceeds through three stages: (a) pre-processing, which includes spell checking, lemmatization, and noise filtering, (b) rule-based pre-checks for particular student response types, and (c) classification of self-explanation quality. The initial pre-check (stage 2) rules identify submission types including copy/pasting, irrelevant responses, and simple paraphrases. Targeted feedback is provided when these response types are recognized, and follow-up automated classifications are not applied. For example, if the system detects close paraphrases, the student receives feedback indicating that the response is too similar to the text content, and suggests the student use

their own words to self-explain (e.g., "That looks very similar to the text. It will help you to understand the text better if you put it in your own words"). When the self-explanation does not trigger any of the pre-checks, it proceeds to the classification algorithm (stage 3). At this stage, the quality of the student's self-explanation is scored from 1 (fair) to 3 (great), and the pedagogical agent delivers formative feedback to improve future self-explanations.

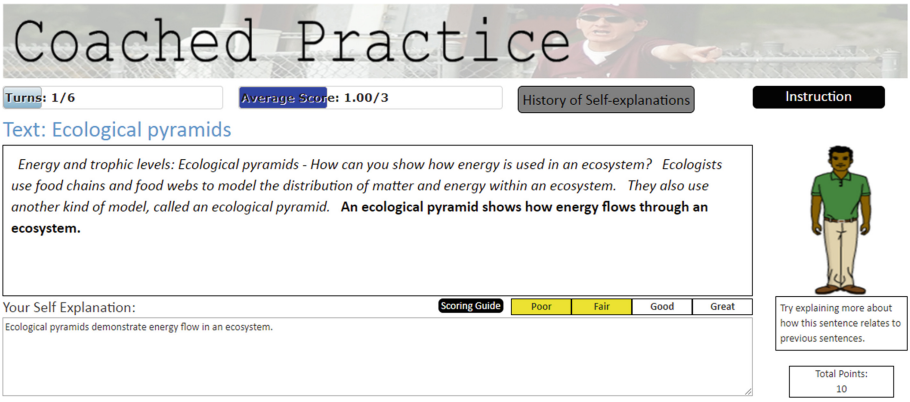

Fig. 1. iSTART's coached practice provides generative self-explanation practice with formative feedback.

3 Method

3.1 Corpus

We collected 4,109 self-explanations using responses from 277 high school participants. Participants completed the self-explanation task for two texts, "Heart Disease" (~300 words) and "Red Blood Cells" (~280 words). They were shown the texts in segments, delimited by target sentences. Each segment of text ended with a target sentence, which the participants were instructed to self-explain: "Please read the text and provide a self-explanation for the **bolded** sentence below." Both texts included 9 target sentences.

Two trained researchers applied the self-explanation coding scheme, which classifies self-explanations from 0 to 3. Self-explanations that were scored 0 (poor) included unrelated, vague, or non-informative information, were too short, or too similar to the target sentence. Scores of 1 (fair) were focused on the target sentence, primarily paraphrasing content from the target. Scores of 2 (good) included 1–2 ideas that were outside of the target sentence. Scores of 3 (great) incorporated information at a global level, tying in information from prior knowledge, or multiple connections across ideas in the text. Before applying the coding scheme to the entire self-explanation corpus, the researchers completed two training rounds on a subset dataset to achieve acceptable interrater reliability. Both raters scored 60% of the full

dataset to obtain 20% overlap for the final interrater reliability (*kappa* = .81). Table 1 shows the distribution of human scores.

Table 1. Distribution of human scores.

Scores	0 (poor)	1 (fair)	2 (good)	3 (great)
# of SEs	124	1,514	1,740	731

3.2 Algorithm Workflow

The main purpose of our comprehensive assessment pipeline is to provide learners with automated scores for their self-explanations ranging from 0 to 3, along with formative feedback as described in the previous section. The algorithm receives as input the self-explanation text from the learner, the target sentence and the previous text and computes an automated score along with relevant feedback for the participant. The algorithm also receives the previous self-explanation as an input in order to check the relevance between successive responses. One consideration in iSTART is that an instructor may enter new texts into the system, and thus the algorithm cannot apply a domain specific model for each new text. This constraint increases the challenge of developing an accurate algorithm. However, this occurs seldom, and most experiments are performed using the existing collection of 24 documents covering general knowledge.

As presented in Fig. 2, the first step of the workflow pre-processes the input text, checking the spelling of words and applying the default NLP processing pipeline from *ReaderBench*. The next step in the process checks a list of rules for detecting different scenarios of poor self-explanations and provides appropriate feedback. The first rule verifies presence of frozen expressions in the student's self-explanation based on predefined lists of regular expressions that match certain conditions (e.g., misunderstanding patterns, prediction, boredom, etc.). If more than 75% of the self-explanation contains frozen expression, then the learner is assigned a 0 score, and an appropriate feedback message is generated (e.g., "Please focus more on the task at hand.").

The second rule checks the semantic cohesion between the participant's explanation and the targeted text; if the cohesion value is below a specific threshold, a poor score is assigned, and relevant feedback provided (e.g., "You should relate more to the given text."). The next criterion considers the length of self-explanation. If the self-explanation is too short when compared with the target text; the student is instructed to add more to the self-explanation (e.g., "Can you please provide more details?"). The threshold for short length was determined using *grid-search* optimization over the SVM hyper-parameters in order to find the optimal parameters in the formula that detects short sentences, as presented later in the Results section. The last rule identifies self-explanations that are copied directly from the target sentence (the text that the student self-explains) or the text prior to the target sentence. This rule uses n-grams from the target text, prior text and self-explanation to perform this copy-related check.

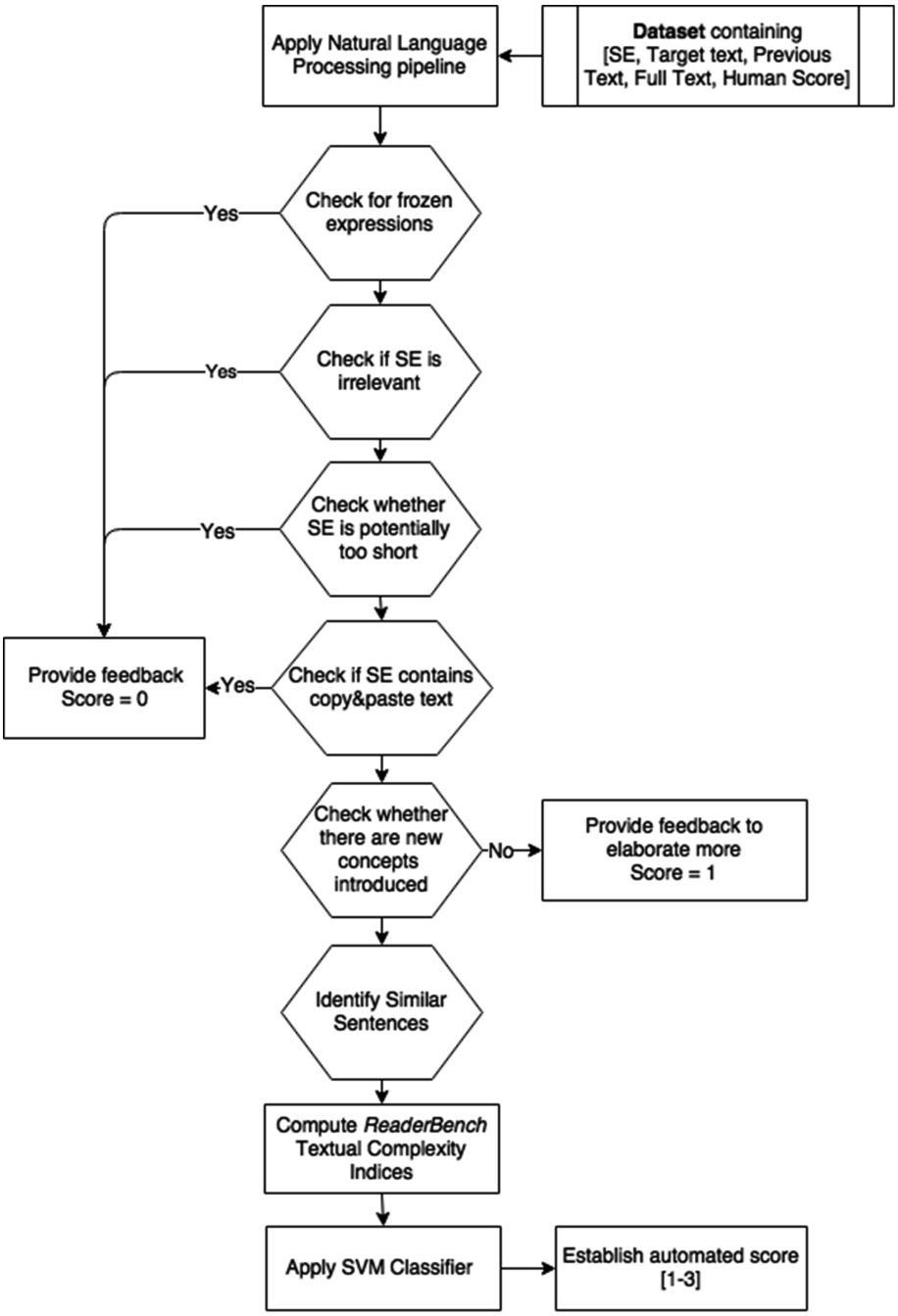

Fig. 2. Workflow for the self-explanation scoring comprehensive tutoring system (including pre-processing, rule-based system and automatic assessment).

Apart from the checks performed to identify low-scoring self-explanations, an additional rule identifies any new concepts that the learner may have introduced while self-explaining the text. In this step, we extract words that are neither identical lemmas nor synonyms of target content words (i.e., non stopwords that have as corresponding part-of-speech one of the following tags: noun, verb, adjective or adverb). In this case, the proposed score for the student is 1 (fair) with paraphrase related feedback to include new concepts - e.g., "Good. Now can you try to explain the text using your own words and ideas".

The last part of the algorithm automatically predicts the self-explanation score as 1 (fair), 2 (good) or 3 (great) using the pre-trained SVM model. The features computed for each self-explanation include: a) the semantic similarities between the self-explanation and target text (including LSA, LDA and word2vec), and b) the textual complexity indices from *ReaderBench*. A nu-SVM [11] algorithm from libsvm Java library [12] was implemented for the prediction model. In this specific implementation of the SVM algorithm, a multi-class "one vs one" strategy is used meaning that a separate classifier is trained for each pair of labels. In addition, ten-fold cross-validation was used for model evaluation, and grid search [11] optimization was performed to tune the parameters for determining the short sentence threshold using parameter variance [0.00; 0.05; 0.10; 0.15; 0.20]. Other hyper-parameters used for training the SVM models are: kernel-RBF (radial basis function), gamma-0.0017 (free parameter of the Gaussian function) and 3 prediction classes. Performance details of the grid search optimization are presented in the results section.

3.3 Encountered Challenges and Envisioned Solutions

Several iterations over the modules were performed before finalizing the format of the algorithm. The iterations included tuning the parameters used in the rule-based system for selecting the most suitable parameters for the SVM model. The first step accounted for evaluation of prediction of score 0 (poor) after passing through the rule-based module. We discovered that some self-explanations were assigned score 0 by the rule-base with "too-short self-explanation" feedback, even though human experts had scored them as 1 (fair). In order to address this issue, the *grid-search* optimization was performed on the full processing pipeline including the SVM, and parameters with the best average prediction were selected. The alternative was using a singular learning model exclusively for the two parameters corresponding to the short length formula, which would not have been globally efficient.

Another challenge encountered in the process was identifying the most relevant parameters from *ReaderBench* textual complexity indices for training the SVM model. The textual complexity parameters from the ReaderBench framework resulted in 1,322 parameters. These textual complexity features were obtained for a dataset consisting of approximately 4,000 SEs presented in the corpus sub-section. Some of these parameters were not relevant for predicting self-explanation scores and could led to over-fitting for the model; as a result these parameters were eliminated using a variance threshold [13] feature selection algorithm on the whole dataset. Elimination of features using variance threshold resulted in 250 features, which were used to train the model.

While designing workflow for an interactive system, it is important that users receive instant feedback. Initial iteration implementing the complete workflow that included pre-processing, the rule-based system, and automatic evaluation using the SVM model, resulted in an average response time of *7 s* for a single SE (measured when running the system on a single machine with 16 GB of memory). We observed that the default NLP processing pipeline in *ReaderBench* accounted for most of this delay. The stages in *ReaderBench* that resulted in this delay included computation of parameters for discourse analysis, dialogism, Cohesion Network Analysis [14] and sentiment analysis.

The best solution to avoid these delays was to eliminate the unnecessary steps from the pipeline and include only the relevant indices for the SVM model to predict the scores. As a result, we computed only those indices that were relevant to the trained dataset – surface, syntax, cohesion, word-complexity, word-list and connectives. In addition, the cohesion graph [14] from the discourse analysis was also retained in the workflow. The average response time for a self-explanation was reduced to 5 s once all these optimizations were performed. In addition, caching for the targeted sentence model was also implemented, but this did not improve the overall response time because students' responses vary greatly and require full processing for each input.

4 Results

This section presents the results obtained after training the SVM model over the 4019 self-explanations from "Heart Disease" and "Red Blood Cells" datasets. In the early stages of training, the short-sentence rule was generating false 0 scores, so we tuned the parameters for calculating a suitable short threshold value (see Eq. 1 inspired from the algorithm implemented in iSTART. for which the threshold values were experimentally set; *noWords* refers to the number of words from the SE).

$$short\ threshold = \begin{cases} [\min(no\ Words, 20) * (0.4 + \Delta x)], \text{if } noWords \text{ from target text} \geq 15 \\ [noWords * (0.5 + \Delta y)], otherwise \end{cases} \tag{1}$$

In order to perform *grid-search* over SVM training, the $\Delta x/\Delta y$ hyper-parameters were varied from 0 to 0.2. During each model-training step, the dataset was divided into train and test using a ten-fold split. For each combination of the *grid-search* parameters, average accuracy was calculated over the test prediction accuracies obtained for all the 10 folds. The results are shown in Table 2. The best accuracy for test dataset is 59% and the corresponding hyper-parameters for short threshold are 45% and 50%. A Kappa of .43 between the algorithm-produced scores and the human rating denotes a moderate agreement.

Table 2. Accuracy for 10-fold SVM with grid search optimization.

$\Delta x/\Delta y$ parameters	0.00	0.05	0.10	0.15	0.20
0.00	.57	**.59**	.56	.56	.56
0.05	.57	.57	.57	.57	.56
0.10	.57	.57	.57	.57	.57
0.15	.56	.56	.56	.56	.57
0.20	.56	.56	.56	.56	.57

We also computed adjacent accuracy, which accounts for the percentage of predicted scores which differ from the human score by less than or equal to 1. The results for adjacent accuracy are given in Table 3 for each stage of the grid search optimization. The adjacent accuracy for the system is very high (97%), demonstrating that the system-predicted scores are close enough to the human experts' scores, though these could be further tuned with additional features to improve exact accuracy.

Table 3. Adjacent accuracy for the 10-fold SVM.

$\Delta x/\Delta y$ parameters	0.00	0.05	0.10	0.15	0.20
0.00	.96	**.97**	.97	.96	.96
0.05	.96	.96	.95	.96	.96
0.10	.94	.94	.96	.96	.96
0.15	.95	.95	.96	.96	.96
0.20	.95	.95	.96	.95	.96

The adjacent accuracy of 97% is also reflected in the confusion matrix obtained over the ten-fold SVM, as well as the evaluation with the best results for the hyperparameters (see Table 4). In contrast to the human scores (see Table 1), the classes with scores of 2 and 3 are more distributed over all the automated score categories (see Table 5).

Table 4. Confusion matrix for the best parameters ($\Delta x = 0.05$ and $\Delta y = 0.00$) of the grid search optimization over the SEs.

SVM score/Human score	1	2	3
1	994	409	108
2	450	915	325
3	115	265	351

Table 5. Distribution of automated scores.

Scores	0 (poor)	1 (fair)	2 (good)	3 (great)
# of SEs	70	1559	1589	784

5 Conclusions

This paper reported the development of an automated self-explanation evaluation workflow, and the challenges encountered when implementing the workflow in the tutoring system. The proposed workflow integrating the textual complexity indices from the *ReaderBench* framework is specifically designed to provide automated comprehensive feedback in iSTART's self-explanation practice. The formative feedback is designed to improve students' comprehension strategy use while the self-explain texts.

iSTART's rule-based evaluation first identifies clear deficiencies, including irrelevant, too short, or copied responses. Using this first level of self-explanation assessment, iSTART's instructional model intervenes to provide targeted feedback to address such deficiencies. We leveraged machine learning algorithms to predict the self-explanation scores and evaluated the resulting model against the human experts scores.

In the final workflow, the rule-based system is used to detect poor self-explanations based on noise, relevancy, length, and similarity that are scored 0, and the SVM trained model for scoring self-explanations from 1 (fair) to 3 (great). Scores resulting from the SVM model also inform iSTART's instructional model, resulting in feedback aimed at improving the overall quality of the student's self-explanation. Despite the fact that the overall accuracy is 59% for the testing set, the adjacency measure of 97% demonstrates that the model classifies majority of the SEs near the targeted class.

During the development of the algorithm workflow, we encountered several problems. One was finding the most relevant and suitable set of features for the SVM model. We opted to select indices that exhibited sufficient linguistic coverage (experimentally set at a minimum 20%) and complement one another by expressing different traits (Pearson correlation lower than .9).

The second major issue faced during implementation of the workflow was response time due to evaluation of the rule-based system and NLP processing pipeline computation from *ReaderBench*. The solution proposed for improving the response time was to compute only the features and textual complexity indices that were necessary for the SVM classification model. We found that the *ReaderBench* framework utilizes a large number of NLP processes such as part-of-speech (PoS) tagging, parsing, dependency parsing, co-reference resolution and named entity recognition (NER) for generating the vast set of linguistic indices. However, the only set of features that were relevant for the task in this study were related to PoS tagging and textual complexity indices. As a result, we eliminated the unnecessary phases (dependency parsing, co-reference resolution, and NER) from the NLP pipeline, thus reducing the processing time to some extent.

However, our sense is that even with all these optimizations, the processing time due to the NLP pipeline remains excessive; waiting 5 s for feedback every time one generates a self-explanation would be excessively annoying. This processing cost currently renders it unfeasible to integrate this algorithm within learning scenarios that require near real-time responses. Nevertheless, this scoring system is highly accurate, nearing the accuracy obtained between two highly trained human scorers; in addition, it is more

efficient for students to obtain immediate feedback for their answers. Hence, we do intend to use the algorithm to provide scores in contexts where system response time is not a critical factor. Moreover, we expect the processing power of computers to continue to increase, rendering the processing time shorter for these types of implementations.

Taken together, this study can provide valuable lessons for the larger NLP and educational research community. First, having a too wide range of textual features without a proper systematic preprocessing of data and appropriate classification rules established by human experts can be cumbersome and exhibit lower accuracies, despite using more advanced automated classification methods. Additionally, NLP researchers must consider the constraints of the particular environment into which the evaluation approach is being implemented. In our case, the response time of the initial workflow was not practical, due to iSTART learners' interaction behaviors and their expectations of the system.

Our next steps will focus on further tuning the model hyper-parameters in order to better differentiate between scores, including *grid-search* optimization over other SVM parameters. Moreover, we can apply more advanced feature selection algorithms, such as L1-based [15] or Tree-Based [16], for improving the prediction of each class. All these improvements can lead to the integration of the pipeline in a near real-time tutoring system that performs comprehensive automated evaluations of student response.

Acknowledgments. This research was partially supported by the 644187 EC H2020 *Realising an Applied Gaming Eco-system* (RAGE) project, the FP7 2008-212578 LTfLL project, the Department of Education, Institute of Education Sciences - Grant R305A130124, as well as by the Department of Defense, Office of Naval Research - Grants N00014140343 and N000141712300. We would also like to thank Tricia Guerrero and Matthew Jacovina for their support in scoring the self-explanations.

References

1. Jurafsky, D., Martin, J.H.: An Introduction to Natural Language Processing Computational Linguistics, and Speech Recognition. Pearson Prentice Hall, London (2009)
2. Dascalu, M., Dessus, P., Bianco, M., Trausan-Matu, S., Nardy, A.: Mining texts, learner productions and strategies with *ReaderBench*. In: Peña-Ayala, A. (ed.) Educational Data Mining. SCI, vol. 524, pp. 345–377. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-02738-8_13
3. Dascalu, M., Dessus, P., Trausan-Matu, Ş., Bianco, M., Nardy, A.: *ReaderBench*, an environment for analyzing text complexity and reading strategies. In: Lane, H.C., Yacef, K., Mostow, J., Pavlik, P. (eds.) AIED 2013. LNCS (LNAI), vol. 7926, pp. 379–388. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39112-5_39
4. Drucker, H., Burges, C.J., Kaufman, L., Smola, A.J., Vapnik, V.: Support vector regression machines. In: Advances in Neural Information Processing Systems, pp. 155–161 (1997)
5. McNamara, D.S., Levinstein, I., Boonthum, C.: iSTART: interactive strategy training for active reading and thinking. Behav. Res. Methods Instrum. Comput. **36**(2), 222–233 (2004)

6. O'Reilly, T., Sinclair, G., McNamara, D.S.: iStart: A Web-Based Reading Strategy Intervention That Improves Students's Science Comprehension. In: CELDA, pp. 173–180 (2004)
7. McNamara, D.S.: SERT: self-explanation reading training. Discourse Process. **38**(1), 1–30 (2004)
8. McNamara, D.S., Magliano, J.P.: Self-explanation and metacognition. In: Handbook of Metacognition in Education, pp. 60–81 (2009)
9. McNamara, D.S., O'Reilly, T.P., Rowe, M., Boonthum, C., Levinstein, I.B.: iSTART: A web-based tutor that teaches self-explanation and metacognitive reading strategies. In: McNamara, D.S. (ed.) Reading comprehension strategies: Theories, interventions, and technologies, pp. 397–420. Erlbaum, Mahwah (2007)
10. Boonthum, C., Levinstein, I., McNamara, D.S.: Evaluating self-explanations in iSTART: word matching, latent semantic analysis, and topic models. In: Kao, A., Poteet, S. (eds.) Natural Language Processing and Text Mining, pp. 91–106. Springer, London (2007). https://doi.org/10.1007/978-1-84628-754-1_6
11. Liu, R., Liu, E., Yang, J., Li, M., Wang, F.: Optimizing the hyper-parameters for SVM by combining evolution strategies with a grid search. In: Huang, D.S., Li, K., Irwin, G.W. (eds.) Intelligent Control and Automation, pp. 712–721. Springer, Heidelberg (2006). https://doi.org/10.1007/978-3-540-37256-1_87
12. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines. ACM Trans. Intell. Syst. Technol. **2**(3), 27:21–27:27 (2011)
13. Cherkassky, V., Ma, Y.: Practical selection of SVM parameters and noise estimation for SVM regression. Neural Netw. **17**(1), 113–126 (2004)
14. Dascalu, M., McNamara, D.S., Trausan-Matu, S., Allen, L.K.: Cohesion network analysis of CSCL participation. Behav. Res. Methods **50**, 1–16 (2017)
15. Gilad-Bachrach, R., Navot, A., Tishby, N.: Margin based feature selection-theory and algorithms. In: Proceedings of the Twenty-First International Conference on Machine Learning, p. 43. ACM (2004)
16. Sugumaran, V., Muralidharan, V., Ramachandran, K.: Feature selection using decision tree and classification through proximal support vector machine for fault diagnostics of roller bearing. Mech. Syst. Signal Process. **21**(2), 930–942 (2007)