Feedback Provision Strategies in Intelligent Tutoring Systems Based on Clustered Solution Spaces

Sebastian Gross[1], Bassam Mokbel[2], Barbara Hammer[2], Niels Pinkwart[1]

[1] Clausthal University of Technology, Department of Informatics
Julius-Albert-Str. 4, 38678 Clausthal-Zellerfeld, Germany
{sebastian.gross,niels.pinkwart}@tu-clausthal.de
[2] Bielefeld University, CITEC centre of excellence
Universitaetsstr. 21-23, 33594 Bielefeld, Germany
{bmokbel,bhammer}@techfak.uni-bielefeld.de

Abstract: Designing an Intelligent Tutoring System (ITS) usually requires precise models of the underlying domain, as well as of how a human tutor would respond to student mistakes. As such, the applicability of ITSs is typically restricted to well-defined domains where such a formalization is possible. The extension of ITSs to ill-defined domains constitutes a challenge. In this paper, we propose the provision of feedback based on solution spaces which are automatically clustered by machine learning techniques operating on sets of student solutions. We validated our approach in an expert evaluation with a data set from a programming course. The evaluation confirmed the feasibility of the proposed feedback provision strategies.

1 Introduction

Intelligent Tutoring Systems (ITSs) have made great strides in recent years. The goal of these educational technology systems is to provide intelligent one-on-one computer-based support to students as they learn to solve problems in a type of instruction that is often not available because of scarce (human) resources. The basic rationale underlying traditional ITS is that by using formalized domain models and dynamic student models, comparing student's problem solutions to the domain models, it is possible to adapt instruction to the needs of specific students and to effectively support their learning.

While leading to interesting results, ITS research faces two challenges: (i) current ITSs have their advantages only in well-structured and comparably narrow domains since they need to judge whether and why a given student answer is correct or wrong; (ii) to achieve this goal, current ITSs require an exact formalization of the underlying domain knowledge which is usually a substantial amount of work: researchers have reported 100-1000 hours of authoring time needed for one hour of instruction [MBA03].

Due to these challenges, most of the ITS research and development has been in domains which are characterized by a well-accepted theory or model that makes it possible unambiguously to decide whether problem solutions are correct or incorrect. Not all domains are that well-defined, indeed most are not. Domains such as law, argumentation, art, or

intercultural competence are characterized by solutions which are ambiguous and cannot be verified formally [LAPA10].

In this paper, we propose a novel method for feedback provision in ITSs. We show that machine learning techniques can be used for structuring spaces of student solutions (for ill-defined and well-defined problems), and that based on these structures, feedback can be given to students. First, in Section 2, we review some relevant existing educational approaches regarding the machine learning of solutions, and we discuss challenges for the corresponding techniques. Next, we discuss feedback strategies based on clustered solution spaces in Section 3. The feasibility of the proposed machine learning techniques and feedback provision strategies is demonstrated in an expert evaluation in Section 4. Finally, we discuss the results and provide an outlook in Section 5.

2 Machine Learning of Solution Spaces

As mentioned above, ITSs in ill-defined domains cannot rely on explicit models which can identify analytically whether and why a student solution is correct/incorrect. As an example, given a programming task, there usually exist many different ways to write a program; because of these ambiguities, often, a semantic error (such as a wrong initialization of a variable) cannot easily be located automatically since alternative solutions where the observed error would actually be correct can exist. For such tasks, it is still possible to gather implicit information in terms of examples given by students or experts. Based on such data, a structure of the solution space can be inferred implicitly. This possibility is widely data-driven, such that machine learning offers a suitable technology to infer meaningful information from given examples for such ITSs.

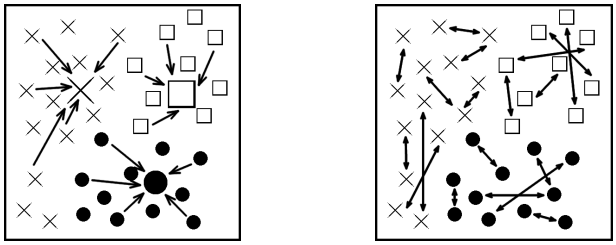
Several machine learning techniques have already been applied successfully to extract information from educational databases. Hierarchical clustering has been applied in student modeling to cluster similar cases into classes to discover high-level student behaviors [RBGL07]. Perera and colleagues [PKK+08] used clustering and data mining techniques in an online collaborative environment to support learning groups by indicating relevant aspects of the groups' operation and providing feedback about where problems are. Su and colleagues [STW+06] proposed a Learning Portfolio Mining approach to extract maximal frequent learning patterns from student's learning sequences. The created clusters were used to assign new learners to a suitable cluster according to their learning characteristics and capabilities. Another recent approach that involves a simple form of machine learning of meta-knowledge in form of action plans has been proposed by Nkambou and colleagues [NFVN10] in the context of an ITS for astronaut training. Here, two knowledge discovery techniques, sequential patterns mining and association rules discovery, were used to find frequent action sequences and association between them. However, the presented knowledge discovery approach is restricted to procedural domains and has not been empirically evaluated yet. Researchers in the field of educational data mining also tackled the challenge to predict students' skills by applying and adapting state-of-the-art techniques for knowledge discovery and clustering, see e.g. [NDA10, TPSH11].

In this paper, we investigate the potential of unsupervised statistical machine learning to

structure the solution space of an ITS used within the frame of programming courses. In this setting, challenges are posed by two facts: (i) Typical data is not given by Euclidean vectors as common in statistical machine learning. Rather, structural properties play an important role, which are hard to express with distinct numerical features. Instead, pairwise comparisons using a problem-adapted similarity metric seem more suitable to take into account complex characteristics of the data. (ii) Models inferred by machine learning should offer an interface towards human-understandable feedback mechanisms. This rules out typical black box techniques which learn a function (e.g. whether a student solution is correct) but which do not explain why they take a decision. We propose to use prototype-based methods (see e.g. [Koh95]) because they fulfill both requirements: they allow human insight since their behavior is based on the notion of similarity and prototypical solutions which can directly be inspected by humans. Recently, prototype-based machine learning has been extended to handle general non-Euclidean data, see [HH10].

Now, we shortly introduce the basics of the clustering algorithm used in our studies. Assume a set of data points $\mathbf{x}^1, \ldots \mathbf{x}^n$ (representing student solutions) is given, whereby the points $\mathbf{x}^i$ might refer to student programs, for example. Assume a dissimilarity $d_{ij} = d(\mathbf{x}^i, \mathbf{x}^j)$ is fixed which assigns a positive value to every pair of data points, indicating their dissimilarity. For example, $d_{ij} = 0$ indicates that two student solutions are equal; a large value indicates that the solutions are very dissimilar. We will later test different available dissimilarity measures which are able to evaluate the dissimilarity of student solutions in an ITS. A prototype-based model represents such data in terms of a small number k of prototypical solutions $\mathbf{w}^1, \ldots, \mathbf{w}^k$. These prototypes decompose the data space into clusters: a prototype $\mathbf{w}^j$ represents all data points $\mathbf{x}^i$ which are closest to it. Thus, decisions why a certain element $\mathbf{x}^i$ belongs to a given cluster can be substantiated by referring to this prototype $\mathbf{w}^j$.

Many different machine learning techniques are currently available to infer representative prototypes given a set of data points [Koh95]. Most techniques have been proposed for data represented by Euclidean vectors. Relational Neural Gas (RNG) constitutes one extension to non-Euclidean dissimilarities [HH10, HH07] of the form d_{ij}, where we assume symmetry ($d_{ij} = d_{ji}$) and normalization of the self-dissimilarities ($d_{ii} = 0$). It aims at finding prototypes which are as representative for their cluster as possible. This aim is formalized mathematically by the objective to place prototypes at positions such that the average dissimilarity within clusters is minimal. Since the data set is given only by the pairwise dissimilarities d_{ij} (assembled in a matrix D), an explicit Euclidean vector space with data points $\vec{x}^i$ is unknown, and thus prototypes cannot be represented as explicit vectors. However, assuming the existence of $\vec{x}^i$, prototypes can be represented implicitly as linear combinations $\vec{w}^j = \sum_i \alpha_{ji} \vec{x}^i$ with $\sum_i \alpha_{ji} = 1$. It was shown in [HH10, HH07] that the iterative adaptation of the prototypes can be realized based only on the known dissimilarities in D, without referring to the (yet unknown) vectors $\vec{x}^i$, and standard numeric methods can be used to optimize this objective. Therefore, training the clustering model results in k representative prototypes of the form $\vec{w}^j = \sum_i \alpha_{ji} \vec{x}^i$. Since training in RNG is based on a numerical optimization procedure involving a random initialization (like in many other clustering techniques), results may differ between different runs, but usually are stable in practice. Note, that in RNG the number of clusters k needs to be fixed a priori, and is usually adjusted w.r.t. prior knowledge about the data in practical applications, exploring different settings. In our experiments in Section 4.2, we used visualizations to



(a) with representative solutions

(b) without representative solutions

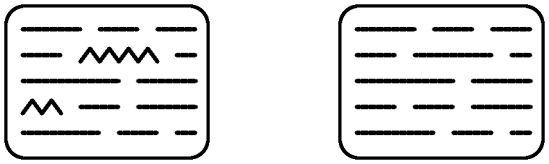
Figure 1: Clustered solution spaces

gain insight into the overall structure of the data, indicating plausible choices for k. For better interpretability of the clustering model, we approximate every prototype by one explicit solution $\vec{x}^i$. Depending on the desired feedback strategies, we can choose the closest available solution, or, alternatively, the closest solution which is correct. In Section 4, we will illustrate the utility of this approach in an expert evaluation.

3 Feedback Strategies Based on Clustered Solution Spaces

The aim of an ITS typically is to support the learning process of students by giving them feedback on their solutions. Feedback can be formed differently (e.g., response accuracy, correct answer, hints, examples) [Mel05] and can be provided at various times during the learning process (e.g., immediately, or after some time has elapsed) [KK88]. For settings where an explicit domain model is lacking, an automatic differentiation between correct and incorrect solutions is hardly possible. This influences the way in which feedback can be provided for students. Lynch and colleagues [LAPA10] identified four human tutoring strategies that can be incorporated into ITSs also for ill-defined problems that lack clear-cut domain models: case studies, collaboration, weak theory scaffolding, and expert review. In this section, we propose feedback provision strategies which are applicable for ITSs in ill-defined domains where feedback is given in the absence of formal domain models – based on data spaces (consisting of student solutions) that are automatically clustered using machine learning as proposed in the previous section.

We base our discussion on clusters of solutions where the solutions within each cluster might have a different quality but are structurally similar. Here, an important difference is whether or not scores (referring to the correctness of a solution) are available for a representative part of cluster elements.



(a) student's solution (b) representative solution

Figure 2: Comparison of solutions

3.1 Feedback provision using representative solutions

In the first case (illustrated in Fig. 1a), we assume that most of the student solutions in the data set have been graded (e.g., by experts). Then, every class of the solution space is represented by a student solution which has a high structural similarity to all other solutions in the cluster (i.e, it is the center of the cluster), and is highly assessed by an expert (i.e., it is a solution which solves the task accurately).

These solutions represent the clusters (shown via large symbols in the figure). They can be used to give feedback to students. Student solutions can be compared structurally to the representative solution, and the result can serve as feedback to students in various forms, including (i) a *direct comparison* contrasting the student's solution and the representative one, emphasizing differences between both, or (ii) the *highlighting* of parts in the student's solution which are potentially erroneous (i.e., the parts where it differs from the representative solution) without explicitly showing the representative solution.

Fig. 2 illustrates a first form of feedback provision: the student's solution is contrasted to the representative solution by displaying both side-by-side and highlighting partial differences, to help the student find mistakes and improve her solution. Typical feedback messages could be formulated as self reflection prompts which have shown to be an effective form of intervention [CBL$^+$89] – e.g., by asking students to explain how their solution differs from the other, and why any of these might be better. In type (ii) feedback provision, a corresponding message would ask students to focus on the highlighted parts of their solution and to explain those parts.

3.2 Feedback provision without representative solutions

In the second case (illustrated in Fig. 1b), we assume that the solutions have not been graded by experts such that no scores are available. Thus, representative correct solutions as defined in the previous subsection cannot be computed. In this case, we propose peer-to-peer interactions as a kind of collaboration between students.

The use of peer-reviewing among the group of students can help a student to improve her solution. The students check the solutions of their peers for correctness and give an assessment in the form of a score and a short comment. Peer reviewing has shown to produce assessment results of good quality as long as 4 to 5 reviews for a solution

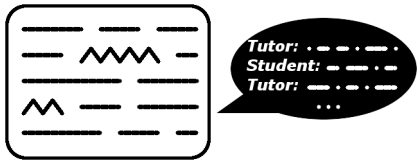


Figure 3: Dialogue-based peer tutoring

are available [CS07]. While peer reviews have the advantage that they build up domain knowledge over time (and thus make the computation of representatives possible), their disadvantage is that results (and thus, feedback) for a newly submitted solution are usually not available instantly.

To avoid this disadvantage, another option is peer tutoring [GH89, MRRT92]. A reviewing student is tutoring another student, so the peer tutor can give hints about evident mistakes and can ask questions about potential mistakes. Fig. 3 sketches a student solution with a dialogue between student and peer tutor. The peer tutor can recommend a change of erroneous parts of the solution. This feedback can help the student to improve her solution. As a side effect, also the peer tutor gets trained in finding and avoiding mistakes.

In this second case, the ITS does not compute feedback directly but acts as a mediator between students. Since the reviewer / peer tutor is a student who is in a learning process herself and does not possess expert knowledge, the ITS has to choose both student and reviewer / peer tutor intelligently. Student and reviewer must be selected in such a way that the student profits by reviewer's tips and the reviewer is not overwhelmed by reviewing student's solution. The clustering of solutions can be helpful here, since it can guarantee that both students have submitted at least a structurally similar (though possibly not equally good) solution.

There also exists the case that for some clusters representative solutions (case 1) are available and for some they are not (case 2). In this case, feedback strategies proposed in case 1 and 2 can be combined to give feedback to students.

4 Validating the Approach: an Expert Evaluation

For validating our approach, an expert evaluation with four experts was conducted. The experts were student assistants who had been tutors in a Java programming class. They were all highly experienced and fully qualified in checking and assessing student solutions.

4.1 Research Questions

Concerning the proposed feedback strategies based on clustered solution spaces (see Section 3), we were interested in the following research questions:

1. Can a comparison of a student solution to a representative solution help a student to improve her solution if (i) a *direct comparison* is used, showing the student solution and the representative solution, highlighting differences between both, or (ii) no direct comparison is used, rather *highlighting* of potentially erroneous parts in the student solution without explicitly showing the representative solution takes place?
2. Does clustering with RNG (see Section 2) assign student solutions to suitable clusters represented by a solution with a high similarity to each solution within the cluster?

We hypothesized that the proposed feedback strategies help students to improve their solutions. We expected that each cluster consists of student solutions with a high similarity to each other.

4.2 Clustered Solution Space: Machine Learning Technique and Metric

We based our evaluation on a data set from a Java programming class (first semester students) at Clausthal University of Technology which consisted of 165 student solutions for the same task. For this data, scores assigned by human experts were available for every student solution. We used the open source plagiarism detection software *Plaggie* [ASR06] to extract a tokenized representation (a *token stream*) from each given Java source code. Based on the token streams, we considered three different (dis)similarity measures to be calculated between all pairs of Java programs: (i) Euclidean distances on the tf-idf weights [Rob04] for the token streams, (ii) the normalized compression distance (NCD) which is a string dissimilarity measure based on the Kolmogorov complexity [CV05], and (iii) *Greedy String Tiling* (GST) which is the inherent similarity measure that Plaggie uses to compare the given sources. The GST measure [PMP00] assesses the agreement between each pair of token streams. It uses a matching heuristics which aims to cover one given token stream with the largest possible non-overlapping subsequences of the other token stream, rating the amount of coverage in relation to the total number of tokens. This pairwise comparison results in a matrix S of similarity values $s(\mathbf{x}^i, \mathbf{x}^j) \in S$. The values were in $(0, 1)$ where all self-similarities (i.e. entries on the diagonal of S) equal 1. We converted S into a dissimilarity matrix by taking $D := \sqrt{1 - S}$ as proposed in [PD05], followed by symmetrization by setting $d_{ij} = d_{ji} := (d_{ij} + d_{ji})/2$ where necessary.

To investigate which of the three measures is most suited for our expert evaluation, we created 3-dimensional visualizations of each dissimilarity data set, using the state-of-the-art dimensionality reduction technique *t-Distributed Stochastic Neighbor Embedding* (t-SNE) [vdMH08]. Fig. 4 shows the result of embedding the data from the GST similarity matrix to three-dimensional vectors using the t-SNE technique. In case of the GST measure, the visualization clearly shows a structure of 4 clusters, which was the most distinctive and noticeable pattern that we observed in comparison with the other visualizations. This separation into 4 clusters was also visible with other standard dimensionality reduction methods which we applied, like *Multi-dimensional Scaling* (MDS) [BG05] for instance.

For further evaluation, we tested the structural features of the three data sets with a simple quantitative analysis. On each data set, we first trained RNG 50 times and chose

	Distances on tf-idf	NCD	GST
$k = 4$	0.0144	0.2114	**0.6781**
$k = 8$	0.0255	0.1778	**0.4065**
$k = 16$	0.0459	0.1402	**0.2272**

Table 1: $s_{\mathsf{inter}}/r_{\mathsf{intra}}$ for all three considered dissimilarity measures, after clustering with RNG, using three different numbers of prototypes k.

the respective best solution (in terms of the cost function that RNG minimizes) to rule out atypical solutions of the training procedure. For each of the three resulting cluster assignments, we then calculated the maximum distance of data within the same cluster (intra-cluster radius r_{intra}), and the minimum distance occurring between data of different clusters (inter-cluster separation s_{inter}). The ratio $s_{\mathsf{inter}}/r_{\mathsf{intra}}$ reflects how distinctly the data is separated, and how well this separation is captured by the given cluster assignment, higher values being better. We performed this procedure, using three different numbers of prototypes in RNG: $k \in \{4, 8, 16\}$. The other RNG parameters were fixed: 200 training iterations, and an initial neighborhood range of $\lambda_0 = \frac{n}{2}$. The results are reported in Table 1. As expected from what we observed in the visualizations, the best values were obtained for the GST measure. Therefore, we chose the dissimilarity matrix from GST for our following investigations since it leads to the clearest structural discrimination.

With regard to the articulate 4-cluster structure observed in the visualization, we fixed $k = 4$ and trained the clustering with RNG and the above-mentioned parameter settings. The result is shown in the t-SNE visualization in Fig. 4. The symbols of the mapped data points mark their cluster assignment found by RNG, which clearly is consistent with the 4-cluster structure present in the visualization. Each larger pentagram symbol marks the representative of a cluster, namely the best-graded student solution which is closest to the respective RNG prototype of the cluster. Since these points refer to actual student solutions, they can be inspected by humans directly. Hence, the obtained clustering can be used for direct feedback: given a student solution, the system can assign it to its closest representative via the GST measure, and feedback mechanisms as discussed in the previous sections may be applied, e.g. the highlighting of structural differences. In particular, we used these representative solutions for the user study presented in the following section.

4.3 Study Design

After a brief introduction, each of the experts checked and assessed student solutions by an evaluation sheet. For every student solution, the experts should answer (i) how they rate the solution, (ii) which of the proposed representative solutions is suitable for a direct comparison, and (iii) how useful the highlighting of potentially erroneous parts is for improving a student's solution. The first question should be answered on a scale from 1 (very poor) to 5 (very good). For the second question, the experts could select from four options: (1) the representative solution for the cluster as found by RNG, (2) a randomly selected solution of the other clusters, (3) both solutions, or (4) none of them. The third question should also be answered on a scale from 1 (not useful) to 5 (useful).

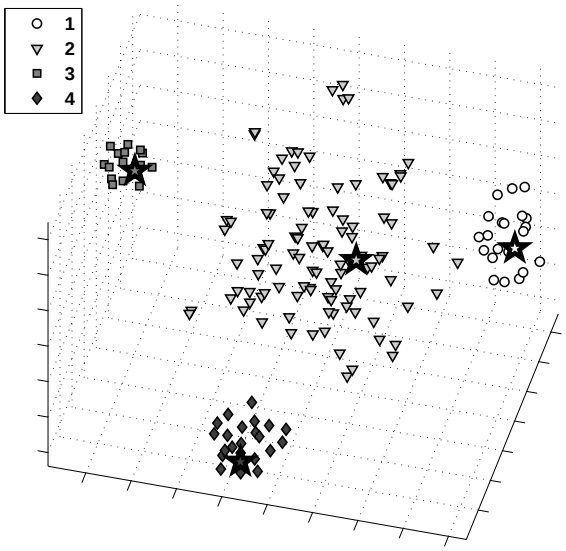


Figure 4: t-SNE mapping of RNG clustering on the similarity matrix from GST with 4 exemplars, each representing a solution with the best grade.

4.4 Results

We first examined whether the experts thought that the provision of feedback by comparing to a representative solution can help students to improve their solutions. Here, the results did not show a clear trend across all of the 161 solutions. Instead, we recognized that the suitability of the feedback provision strategy depends on the rating of the solution. Thus, we performed all analyses depending on the quality of the student solution.

While comparing very good solutions to a representative solution (see Section 3.1 (i)) was not rated as valuable in most cases, the experts stated that (very) poor solutions can beneficially be compared to both suggested representative solutions. For solutions that are already of a good quality, a comparison to a representative solution does not make much

		representative solution			
n	rating	of the same cluster (1)	of a different cluster (2)	both (3)	none (4)
12	very poor	0	3	9	0
15	poor	4	4	7	0
16	average	1	7	5	3
40	good	15	11	6	8
78	very good	11	15	11	41
161	-	31	40	38	52

Table 2: Selected options for feedback provision strategy *direct comparison*

n	rating	average score
12	very poor	4.83
15	poor	4.20
16	average	3.44
40	good	3.1
78	very good	1.95

Table 3: Average score for feedback provision strategy *highlighting*

```
switch (status) {
  case 1:
    if (income <= 8375)
      f = income * (10/100);
```

(a) student's solution

```
switch (statusINT) {
  case 1:
    if(incomeINT <= 8375)
      tax = incomeINT * 0.1;
```

(b) representative solution

Figure 5: Comparison of solutions

sense, because the solution cannot be improved.

The poorer the rating of the solution, the better the feedback strategy *highlighting* was assessed by the experts. For very poor solutions the feedback strategy was highly scored with 4.83. Fig. 5 shows an example how a student can improve her solution by highlighting differences from a representative solution. The student's solution contains a variable of **integer** data type (10/100) representing a fraction, thus it is always zero. In the representative solution, the student used the data type **double** (0.1) instead of integer. By highlighting this part in which the two solutions differ, the student with the erroneous solution could potentially learn that she has used the wrong data type.

5 Conclusion and Outlook

In this paper, we argued that one possible way of providing ITS feedback in the absence of explicit formal domain models is to use machine learning to structure the solution space (as defined by a set of student solutions for a task), and to exploit the structure of this space for feedback provision. Using RNG and a problem-specific similarity measure, we evaluated our approach with a data set from the domain of programming.

The evaluation confirmed our expectations concerning the proposed feedback provision strategies. *Highlighting* of potentially erroneous parts in students' solutions makes sense according to the experts, particularly for solutions which are assessed (very) poor. Also, for (very) poor solutions, representative solutions were considered suitable for a *direct comparison*.

Apparently however, the clustering algorithm did not always assign student solutions to suitable representative solutions (exemplars). The experts chose the closest representative solution (as selected by the algorithm) in 31 out of 70 cases only. We see three possible explanations for this: First, the chosen base metric for the clustering might not have been fully suited for representing relevant semantics of the considered programming task. Sec-

ond, the structure inferred by the clustering algorithm or, alternatively, the given task might have been too simple so that the found representative prototypes do not differ enough from each other, resulting in a different assignment of solutions to prototypes by experts. Third, the experts may (for educational reasons, such as showing the variety of solution options) in some cases have simply have stated that both representative solutions would be suitable for giving feedback.

In future research, we will investigate these problems in more detail, on the one hand by providing feedback based on the closest known correct solution instead of a very limited number of representatives only, and on the other hand by referring to a more powerful similarity metric for analysing and comparing the structure of the programs.

Despite these limitations, the study presented in this paper has shown that feedback provision strategies based on similarity of solutions to known cases and, in particular, feedback by highlighting differences can be a suitable strategy to deal with incorrect student solutions. Our future work will also include empirical aspects, particularly a further validation of the approach with data sets from different domains. Additionally, we will test if the methods of feedback provision can effectively lead to better student learning.

Acknowledgement

This work was supported by the German Research Foundation (DFG) under the grant "FIT – Learning Feedback in Intelligent Tutoring Systems." (PI 767/6 and HA 2719/6).

References

[ASR06] A. Ahtiainen, S. Surakka, and M. Rahikainen. Plaggie: GNU-licensed source code plagiarism detection engine for Java exercises. In *Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling 2006*, Baltic Sea '06, pages 141–142, New York, NY, USA, 2006. ACM.

[BG05] I. Borg and P. J. F. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer, 2005.

[CBL+89] M. T. H. Chi, M. Bassok, M. W. Lewis, P. Reimann, and R. Glaser. Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science*, 13(2):145–182, 1989.

[CS07] K. Cho and C. D. Schunn. Scaffolded writing and rewriting in the discipline: A web-based reciprocal peer review system. *Comput. Educ.*, 48:409–426, April 2007.

[CV05] R. Cilibrasi and P. Vitányi. Clustering by Compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545, April 2005.

[GH89] S. Goodlad and B. Hirst. *Peer tutoring: a guide to learning by teaching*. Kogan Page, 1989.

[HH07] B. Hammer and A. Hasenfuss. Relational Neural Gas. In J. Hertzberg, M. Beetz, and R. Englert, editors, *30th Annual German Conference on AI, KI 2007*, volume 4667 of *LNAI*, pages 190–204, Berlin, 2007. Springer.

[HH10] A. Hasenfuss and B. Hammer. Topographic Mapping of Large Dissimilarity Datasets. *Neural Computation*, 22(9):2229–2284, 2010.

[KK88] J. A. Kulik and C. Kulik. Timing of Feedback and Verbal Learning. *Review of Educational Research*, 58(1):79–97, 1988.

[Koh95] T. Kohonen. *Self-organizing maps*. Springer, 1995.

[LAPA10] C. Lynch, K. D. Ashley, N. Pinkwart, and V. Aleven. Concepts, Structures, and Goals: Redefining Ill-Definedness. *International Journal of Artificial Intelligence in Education*, 19(3):253 – 266, 2010.

[MBA03] T. Murray, S. Blessing, and S. Ainsworth, editors. *Authoring Tools for Advanced Technology Learning Environments*. Kluwer Academic Publishers, Dordrecht, 2003.

[Mel05] E. Melis. Choice of Feedback Strategies. In Kinshuk, D. G. Sampson, and P. Isaias, editors, *Cognition and Exploratory Learning in the Digital Age (CELDA 2005)*. IADIS, iadis, 12 2005.

[MRRT92] D. C. Merrill, B. J. Reiser, M. Ranney, and J. G. Trafton. Effective Tutoring Techniques: A Comparison of Human Tutors and Intelligent Tutoring Systems. *Journal of the Learning Sciences*, 2(3):277–305, 1992.

[NDA10] R. Nugent, N. Dean, and E. Ayers. Skill Set Profile Clustering: The Empty K-Means Algorithm with Automatic Specification of Starting Cluster Centers. In R. S. J. de Baker, A. Merceron, and P. I. Pavlik Jr., editors, *EDM*, pages 151–160. www.educationaldatamining.org, 2010.

[NFVN10] R. Nkambou, P. Fournier-Viger, and E. M. Nguifo. Learning task models in ill-defined domain using an hybrid knowledge discovery framework. *Knowledge-Based Systems*, 24(1):176–185, 2010.

[PD05] E. Pekalska and B. Duin. *The Dissimilarity Representation for Pattern Recognition. Foundations and Applications*. World Scientific, 2005.

[PKK+08] D. Perera, J. Kay, I. Koprinska, K. Yacef, and O. Zaiane. Clustering and sequential pattern mining of online collaborrative learning data. *IEEE Transactions on Knowledge and Data Engineering*, 6(21):759–772, 2008.

[PMP00] L. Prechelt, G. Malpohl, and M. Phlippsen. JPlag: Finding plagiarisms among a set of programs. Technical report, Fakultät für Informatik, Universität Karlsruhe, March 2000.

[RBGL07] V. Robinet, G. Bisson, M. Grodon, and B. Lemaire. Induction of high-level behaviors from problem-solving traces using machine learning tools. *IEEE Intelligent Systems*, 22(4):22–30, 2007.

[Rob04] S. Robertson. Understanding Inverse Document Frequency: On theoretical arguments for IDF. *Journal of Documentation*, 60(5):503–520, 2004.

[STW+06] J.-M. Su, S.-S. Tseng, W. Wang, J.-F. Weng, J. T. D. Yang, and W.-N. Tsai. Learning portfolio analysis and mining for SCORM compliant environment. *Educational Technology and Society*, 9(1):262–275, 2006.

[TPSH11] S. Trivedi, Z. Pardos, G. N. Sárközy, and N. T. Heffernan. Spectral Clustering in Educational Data Mining. In M. Pechenizkiy, T. Calders, C. Conati, S. Ventura, C. Romero, and J. C. Stamper, editors, *EDM*, pages 129–138. www.educationaldatamining.org, 2011.

[vdMH08] L. van der Maaten and G. Hinton. Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.