Scaffolding open text input in a scripted communication skills learning environment

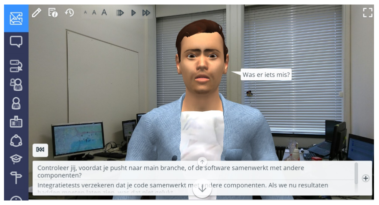
Raja Lala[1], Johan Jeuring[1,2], and Marcell van Geest[1]

[1]Utrecht University, Computer science department, Netherlands
[2]Faculty of Management, Science and Technology, Open University Netherlands

Abstract. Serious games, as well as entertainment games, often employ a scripted dialogue for player interaction with a virtual character. In our serious game Communicate, a domain expert develops a structured, scripted scenario as a sequence of potential interactions in an authoring tool. Communicate is widely used and several domain experts have already developed over a thousand scenarios. In the original version of Communicate, a student 'navigates' a dialogue with a virtual character by clicking one of the multiple statement options at a step of a scenario. Open text response often requires more complex thinking from a student. In this paper we explore ways to handle open text input from a student at a step of a scenario. Our goal is to match open text to scripted statements using a Natural Language Processing (NLP) method and explore mechanisms to handle matched and unmatched input.

1 Introduction

Communication skills are best learned through practice, in role play or with a simulated patient [2]. In Communicate [6], a serious game for training communication skills, a student practices a communication skills dialogue with a virtual character, see Fig. 1. Communicate is used in multiple domains to practice diverse communication skills and protocols, including assertiveness training, breaking bad news, visit to a pharmacy and collaboration.



Controleer jij, voordat je pusht naar main branche, of de software samenwerkt met andere componenten?

Integratietests verzekeren dat je code samenwerkt met andere componenten. Als we nu resultaten

Fig. 1. Communicate game

Authoring content in an Intelligent tutoring system often requires significant effort [1], and authoring tool usability is often at the expense of expressiveness [13]. Communicate provides an authoring tool that combines expressive dialogue constructs with ease of use [7] and runs in a web browser. We release a dialogue scenario editor authoring tool as open-source as part of an EU project RAGE (Realising an Applied Gaming Ecosystem), see `gamecomponents.eu`. A communication skills teacher, usually a non-programmer, authors a scenario. A scenario is a sequence of interleaved subjects, where each subject is a directed acyclic graph consisting of a sequence of statements alternating between a virtual character and a player. A learning goal is typically encoded as a parameter in a scenario. An author assigns values to this parameter, typically an integer, per player statement option. An author also assigns an emotion to a player statement.

Communicate [6] presents statement options to a player at a step of a communication scenario, see Fig. 1. Choosing a player statement elicits the emotion assigned to the statement from a virtual character. At the end of a simulation, a player gets a score depending on her statement choices during the simulation. Martinez et al. [11] review cumulative research on the cognitive demand of multiple choice versus constructed response test item formats. Test item formats pose trade offs in cognitive demand, psychometric characteristics, and costs of administration/scoring. Multiple choice items often elicit low level cognitive processing from a student but are deterministic and easy to score. Open text response often requires complex thinking, but is more difficult to score. Students consider multiple choice fair, but they pay more attention to content when preparing for an open response test.

Our goal is to enhance Communicate by offering a student the possibility to enter open text at a step of a scenario. Adding an input box for a player to enter open text to our simulation is trivial, the challenge is to process this open text. To process open text, we use Natural Language Processing (NLP) techniques. In a previous experiment we gathered student open text input for a scenario and created a golden dataset on which we ran a range of open source NLP methods [9]. NLP methods that use local information (e.g. string kernels) give better matching results than NLP methods that use a generic corpora (e.g. semantic matching using latent semantic analysis, or paraphrasing). Even with a sizeable dataset, the results of NLP methods are not entirely accurate: NLP methods often require very large datasets to train a model [15]. It is unlikely we will obtain large datasets for all scenarios in Communicate: since authors can easily create and/or modify a scenario, we have over a thousand (variants of) scenarios about different communication protocols in different contexts. Our main contribution in this paper is to introduce open text input without significant additional effort from an author and/or extensive data gathering for a scenario, which is infeasible given the number of scenarios, and to explore mechanisms to handle matched and unmatched input. There is a wider implication for entertainment and learning games using scripted dialogues.

Realdon et al. [14] suggests a scaffolding structure in a learning environment to give a student an opportunity to learn. Given the limitations mentioned in the previous paragraph, our approach is to use an NLP method to match a student open text input and use scaffolding to handle matched and unmatched input. This NLP method takes a scripted scenario as input to build a scenario specific corpus [8]. For matching an open input text, this method uses the scenario specific corpus and returns a match score per scripted statement at a step of the simulation. We established a threshold score for the NLP method [9] and if all match-scores are below this threshold, we consider an open text input as unmatched. If at least one match-score is above the threshold, we consider an open text input as matched. Fowler and Barker [3] find that highlighting improves retention of material and to handle matched input, we highlight the best matching statement. To handle unmatched input, we look at Intelligent Tutoring Systems (ITS). Van Lehn [16] studies common behaviour of ITSs and recommends giving a hint for a next step when a student needs a hint. He recommends sequencing hints, starting with encouraging a student to think herself, and after that giving more detail about a next step. Our research questions (RQs) are: Can we handle matched input by highlighting a statement and unmatched input by providing a sequence of hints? We introduce variations in blended teaching sessions to answer our research questions.

This paper is organised as follows. Section 2 discusses related work. Section 3 describes our method to answer the research questions. Section 4 discusses results and Section 5 presents conclusions and future work.

2 Related work

This section gives related work on introducing natural language input in a learning environment or serious game. We look at different approaches to introduce open text input.

Autotutor [4] is a well known tutoring environment using natural language technologies. In AutoTutor, a student answers a question by means of a paragraph (approximately seven sentences) of text. Autotutor provides feedback on this text, and engages in a turn taking dialogue until a student arrives at a number of correct sentences. To handle open text, AutoTutor uses NLP methods like latent semantic analysis and speech act classifiers; techniques that focus on the general meaning of the input and functional purpose of an expression but also require a large dataset. It is unclear how AutoTutor processes unmatched input that does not fit any classifier in the script. The time and cost of authoring a script is considerable and requires extensive collaboration between computer scientists, cognitive psychologists and content experts.

Lessard [10] investigates the design of a natural language game conversation (built in ChatScript) based on experience from three digital games primarily involving a dialogue between a player and a virtual character. An NLP interaction provides creative conversational play, role playing, content contribution and non-linear conversations in these games. The drawbacks mentioned are: a player expectation that the system understands and responds like a person, leaky fictional coherence, unrestricted input, i.e. a player can say anything and a virtual character cannot have a response for everything and 'amnesia' specific to a chatbot that cannot 'remember' previous utterances.

Higashinaka et al [5] conduct an experiment to collect question answer pairs from users to create a chatbot character with consistent personality. The authors use a text engine to index the collected question answer pairs. They develop a chatbot that takes an open text input as a query to retrieve the most relevant question and responds with the pair answer. They use different retrieval methods and perform a 'subjective' evaluation to rate an answer in terms of naturalness and consistency. Higashinaka et al. find that the chatbot character has a consistent personality but the text retrieval methods are not entirely accurate.

(In)accuracy of an NLP method is related to our work; we use scaffolding to handle an open text input.

Min et al [12] present a multimodal framework that predicts breakdowns in a student conversation with a pedagogical agent. The authors characterise a dialogue breakdown as a situation in which an agent misinterprets a student utterance and responds incorrectly. The framework incorporates natural language, eye gaze, student gender, and task state. Min et al. investigate this framework in a study of 92 middle school students in a game based learning environment. They find that incorporating eye gaze achieves high predictive accuracy for dialogue breakdown. We give a sequence of hints to a player when an open text input cannot be matched to a scripted statement.

In summary, there is a diversity of methods to introduce open text input in serious games and learning environments. Methods range from collecting and classifying input, semantic matching, chatbots, and agent technology. In our approach we use a scripted scenario as the basis, which ensures consistency, fictional coherence and control in utterances. We use an NLP method that takes a scripted scenario as input to build a scenario specific corpus [8]. We match an open text input using this corpus and use scaffolding to handle matched and unmatched input.

3 Method

To answer our research question, we introduce variations in our teaching intervention. At our University, final year computer science students learn to work together in a software project team. During the project we provide a communication skills blended learning session per team consisting of 10 to 12 students each. In this session, students play a scenario about collaboration. A student needs to converse with a team mate (virtual character) who has not followed quality procedures (integration tests). In a session, an instructor introduces Communicate, students play the scenario, the instructor explains the communication protocol that forms the basis of the scenario, and there is a plenary discussion. To enable open text, we provide a text box at each step of a scenario, see Fig. 2.

We added hints to the Collaboration scenario that we use in the sessions. For example, we added the hint *'Try to give feedback about working together: his code does not work with other components, so you know that he has not performed integration tests'* to the subject *'Express'*, and we added the hint *'verbalise behaviour'* to the player statement, *'You pushed code that does not work with other components'*.

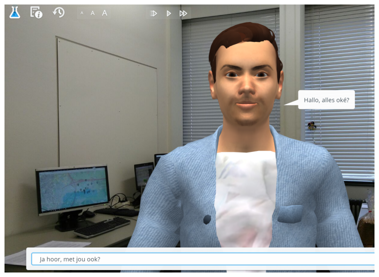


Fig. 2. Open text input box

Our research method is to vary the aspect under consideration. For highlighting our treatment is to let a student match her open text input to a statement option with and without highlighting in separate rounds of sessions respectively. For hints, the treatment is to divide the students randomly into an experimental and control group, where within the same session, a student from the experimental group gets a sequence of hints and the control group get no hints. The design of the sessions in the semester of fall-winter 2018 is shown schematically in Fig. 3.

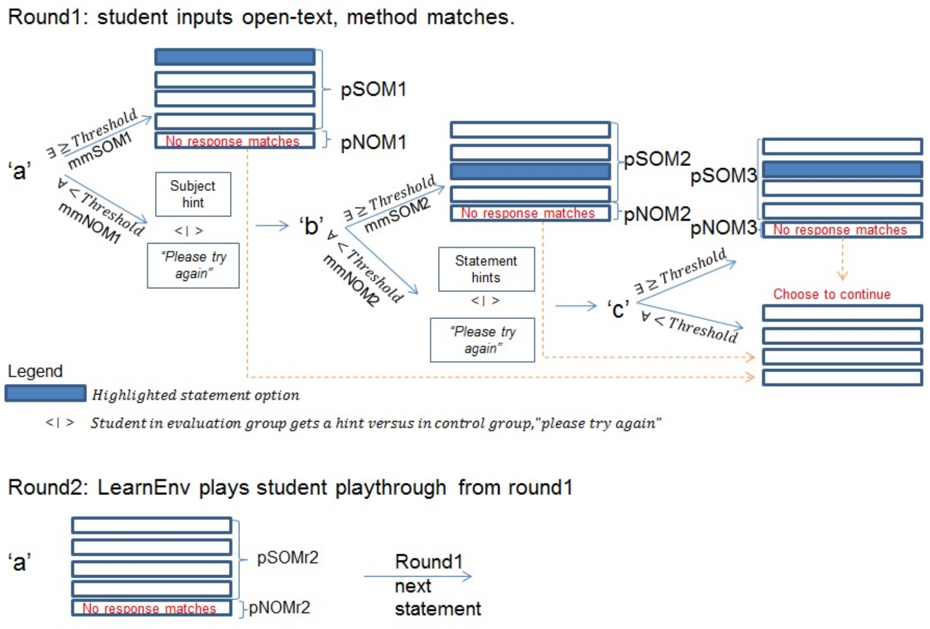
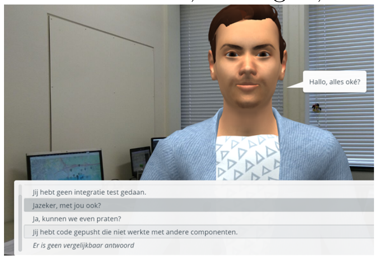


Fig. 3. Setup within sessions of fall-winter 2018

A student fills in an open input text (see Fig. 2, for example 'Ja hoor, met jou ook?') and the NLP match method takes this open input text 'a' and returns a match score per scripted statement option at a step of the scenario. In a previous experiment [9], we empirically determined a baseline threshold value for a match. If at least one statement option has a match score above the threshold value of the match method, we call that an mmSOM (match method some option matches). If a match method detects at least one mmSOM (∃ arrow upwards), Communicate displays all scripted statement options and highlights the best mmSOM match, see Fig. 4, in this example 'Jazeker, met jou ook?'.



Jij hebt geen integratie test gedaan.
Jazeker, met jou ook?
Ja, kunnen we even praten?
Jij hebt code gepusht die niet werkte met andere componenten.
Er is geen vergelijkbaar antwoord

Fig. 4. Highlighted best match

Communicate asks a student to match the closest option to her open input text 'a' from the displayed statement options. If a student selects one of the statement options, we log for statistical comparison (see Section 4) as 'pSOM1' (player some option matches). We also log whether a student selects the highlighted option or another option. A student also gets an option to choose 'No response matches' (*Er is geen vergelijkbaar antwoord*). If a student chooses this option, we log this as 'pNOM1' (player no option matches). If a student finds that no option matches, Communicate asks the student to select one of the scripted statement options to continue the scenario, shown in Fig. 4 as the dotted line from the upper left part in round1 to the lower right part of round1.

If all NLP match method scores for an input statement 'a' are below the threshold value (∀ arrow downwards) of the match method, we say that the open text input is unmatched at that step in the scenario. Note that an open text input is either matched or unmatched using the NLP method. We log an unmatched input as mmNOM (match method no option matches). If the student is in the experimental group, Communicate gives a first hint and prompts to try again. This hint is a subject hint. If the student is in the control group, Communicate gives no hint and prompts a student to try again. A student enters a new response 'b', which is matched to obtain an mmSOM or mmNOM. In case the match method finds an mmSOM, we process it in the same way as described in the paragraph above. In case a match method detects an mmNOM again, Communicate displays the hints for all the statement options at that step, e.g: *Try to: verbalise behaviour; refer to agreement; ask a question* for the experimental group. For the control group Communicate again gives no hint and prompts a student to try again. A student enters a new response 'c', which again is matched to obtain an mmSOM or mmNOM. In case of an mmSOM, we again process as described above. In case of an mmNOM, Communicate asks a student to select one of the scripted statement options to continue the scenario.

For the treatment with no highlighting, we conduct a 2nd round of sessions with the students, three weeks after the 1st round. For all students who agreed in their user profile to store their open text input, Communicate presents a student her first playthrough from the 1st round. At each step of the playthrough, the first open text a student entered (the string 'a') in the 1st round is displayed, and Communicate displays all the statement options available at that step of the scenario and the special option *No response matches.* No statements are highlighted unlike in round1. There are a total of 210 open text input statements that students match. Communicate asks a student to match her input (string 'a') to an option, and we log if a student matches to a scripted option or if she chooses *No response matches.* Communicate displays the match the student made in the first round and continues with the next entered input from her playthrough from the 1st round until the end of the playthrough. Other students, who did not agree to store their open text input, played the scenario in multiple choice format.

4 Results and discussion

In the fall-winter semester 2018, there were a total of 52 students in five project teams, who played the same scenario in a modified version of Communicate where they typed in open text responses. Our research question is whether we can handle matched and unmatched open text input using highlighting and hints respectively. In the majority of open text input in round1 (389 statements out of 503 statements, 77.34%), the match method matched with at least one of the scripted statements. For the remaining statements (114 of 503 statements, 22.66%) for which no match was found: Communicate showed a hint to students in the experimental group for approximately half the unmatched cases (56 statements, 11.13%) versus no hint in the control group (58 statements, 11.53%).

RQ: Can we handle matched input by highlighting a statement?

We compare the match choice from a student in the first round (best matched statement highlighted) versus the second round (no statement highlighted). In the first round, for an mmSOM, a player gets to match her open text input to one of the scripted statements while the best match is highlighted, see Fig. 4. She can choose either the highlighted statement, another statement, or *No response matches*. We analyse the different combination cases that occur. We discuss how a simulation without a highlighting scaffolding (referred as automatic match) would look like. We also gather insight into NLP matching versus student matching. All percentages in this subsection are from a total of 210 open text input statements that students match in round 2, see Section 3. The comparison between round1 (highlighting) versus round2 (no highlighting) is summarised in Table 1.

Does highlighting increase the chance of matching? We argue that highlighting had an effect when a student matches the highlighted statement in round1 and to a different statement; or to *No response matches* in round2. This occurs in 16 (7.62%) and 3 (1.43%) statements respectively, total 9.05% of the statements (210) matched in the 2nd round, shown in the first row of Table 1.

	%
Highlighting effect cases	09.05%
True Positive (TP)	17.14%
True Negative (TN)	07.14%
False Negative (FN)	16.19%
False Positive (FP)	01.90%
NLP match differs from student match	20.48%
Student chooses inconsistently	28.10%

Table 1. Highlighting comparison table

When a player matches her open input with the match method highlighted statement choice in round1 and chooses the same statement (unhighlighted) in round2, we call this a true positive (TP), 36 statements (17.14%) are TPs, shown in the second row of Table 1. When a player chooses *No response matches* in round2 and the match method also detects no match (mmNOM) in round1, we call this true negative (TN), 15 statements (7.14%) are TN, shown in the third row of Table 1. A TN open text input might be a signal of a missing scripted player statement option at a step of a scenario. In an automatic match, Communicate response would be accurate in these (TP and TN) cases. When a student matches her input to a statement in round2 but the match method detects no match in round1, we call this a false negative (FN, 34 statements, 16.19%). In an automatic match, Communicate would incorrectly provide a hint for an FN. When a student chooses *No response matches* in both rounds but the match method finds at least one match value above the threshold, we call this a false positive (FP, only in 4 statements, 1.90%). For an FP in an automatic match, a virtual character would provide a response, but Communicate should have given a hint.

When the NLP match method matches differently than a student statement match in both rounds (43 statements out of 210, 20.48%), in an automatic match a virtual character would provide a different response than possibly intended by the scenario author. E.g. a student entered, 'De code die je gisteren hebt gepushed conflicteert' for which the NLP method has the best match score to 'Ik wil het even met jou hebben over je werk van gisteren' whereas the student matches to 'Je code werkte niet samen met het geheel'. In this example, the NLP choice is not incorrect, however these statements are opportunities to examine if we can improve the NLP match method further.

There are cases when a student chooses inconsistently: when a student matches to an unhighlighted statement in round1 but chooses a different statement in round2 (43 statements out of 210, 20.48%), coincidentally the same amount but other statements than the statements that the NLP method matches differently than a student. We examined these statements: sometimes an input was a mix of two scripted statements or perhaps the statement options seemed similar to a student. Another inconsistency is when a student chooses *No response matches* in one of the rounds and an unhighlighted option in the other round (16 statements, 7.62%). In automatic processing these cases (total 28.10%, shown in the last row of Table 1), either a virtual character's response or a hint would be somewhat correct.

To answer our research question: using highlighting for matched open text input is not effective. The total of absolute errors in matching: false negative and false positive (18.09%) is limited and we argue on the basis of our results that for a matched open input, Communicate should not use highlighting, and automatically continue with the simulation.

RQ: Can we handle unmatched input by providing a sequence of hints? We evaluate if giving a hint in the experimental group leads more often to a matched input than in the control group.

	Control group	Hint group
Initial unmatched statements (mmNOM)	58	56
Observed matches after subject hint	24	28
Expected matches after subject hint	26.4561	25.5439
Unmatched statements after subject hint	34	28
Observed matches after statement hints	16	9
Expected matches after statement hints	13.7097	11.2903

Table 2. Hint evaluation table

We perform a chi-distribution test, which compares the observed cases versus expected values, see Table 2. The experimental (hint) group matches slightly better than expected after a subject hint and slightly worse than expected after a statement hint. The control group matches the other way around, slightly worse than expected after the first prompt to try again and slightly better than expected after the second prompt to try again. The differences are not significant (p-value 0.2521), and the reasons could be multifold. We paid attention while scripting the hints that a hint would not result in a match by copypasting. Perhaps a student tried the same words as in a hint and it could be that a student is perhaps frustrated by having to type something again. The match method is also not entirely accurate, perhaps a hint is similar to a student input which was incorrectly unmatched. To answer our research question, giving a hint for unmatched open text input has no significant impact in our experiment. We recommend to not give hints, but instead to display the available statement options immediately to allow a player to continue a simulation.

5 Conclusions and future work

In this paper we take steps to enhance our learning environment from a multiple choice player input to open text player input. Enabling player open text input in our learning environment leads to more student interaction. Scaffolding, highlighting matched open text input and giving hints for unmatched input, has only a limited effect in Communicate. This result can perhaps be generalised for serious games that use a dialogue graph, want to incorporate open text input and have no extensive dataset. Our experiment results in a dataset of open text matched to a statement annotated by a student herself. The total of absolute matching errors by our NLP method on this dataset is small. The dataset provides a good distribution of student open text with corresponding matching and can be used by other NLP methods to improve match accuracy.

For future work, we recommend and plan to have guided sessions with minimal scaffolding, where in case of matched input a simulation continues as if a virtual character has understood the input (i.e. no extra highlight step to confirm a match) and in case of unmatched input, we present the available statement options to a player to select and continue a dialogue. This setup will involve no extra effort for a scenario author.

6 Acknowledgments

This activity has partially received funding from the European Institute of Innovation and Technology (EIT). This body of the European Union receives support from the European Unions Horizon 2020 research and innovation programme. The authors acknowledge Gemma Corbalan and Matthieu Brinkhuis for their help in the statistical analysis.

References

1. Vincent Aleven, Jonathan Sewall, Bruce M McLaren, and Kenneth R Koedinger. Rapid authoring of intelligent tutors for real-world and experimental use. In *Sixth IEEE International Conference on Advanced Learning Technologies (ICALT'06)*, pages 847–851. IEEE, 2006.
2. Marianne Berkhof, H. Jolanda van Rijssen, Antonius J.M. Schellart, Johannes R. Anema, and Allard J. van der Beek. Effective training strategies for teaching communication skills to physicians: An overview of systematic reviews. *Patient Education and Counseling*, 84(2):152–162, 2011.
3. Robert L Fowler and Anne S Barker. Effectiveness of highlighting for retention of text material. *Journal of Applied Psychology*, 59(3):358, 1974.

4. Arthur C Graesser, Shulan Lu, George Tanner Jackson, Heather Hite Mitchell, Mathew Ventura, Andrew Olney, and Max M Louwerse. Autotutor: A tutor with dialogue in natural language. *Behavior Research Methods, Instruments, & Computers*, 36(2):180–192, 2004.
5. Ryuichiro Higashinaka, Masahiro Mizukami, Hidetoshi Kawabata, Emi Yamaguchi, Noritake Adachi, and Junji Tomita. Role play-based question-answering by real users for building chatbots with consistent personalities. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 264–272, 2018.
6. Johan Jeuring, Frans Grosfeld, Bastiaan Heeren, Michiel Hulsbergen, Richta IJntema, Vincent Jonker, Nicole Mastenbroek, Maarten van der Smagt, Frank Wijmans, Majanne Wolters, and Henk van Zeijts. Communicate! — a serious game for communication skills —. In *Proceedings EC-TEL 2015: Design for Teaching and Learning in a Networked World: 10th European Conference on Technology Enhanced Learning*, volume 9307 of *LNCS*, pages 513–517. Springer, 2015.
7. Raja Lala, Johan Jeuring, Jordy van Dortmont, and Marcell van Geest. Scenarios in virtual learning environments for one-to-one communication skills training. *International Journal of Educational Technology in Higher Education*, 14(1):17, May 2017.
8. Raja Lala, JT Jeuring, FPM Heemskerk, Marcell Van Geest, Jordy Van Dortmont, Gabriel Gutu, Stefan Ruseti, Mihai Dascalu, Beatrice Alex, and Richard Tobin. Processing open text input in a scripted communication scenario. In *SEMDIAL 2018: The 22nd workshop on the Semantics and Pragmatics of Dialogue*, pages 211–214, 2018.
9. Raja Lala, Marcell van Geest, Stefan Ruseti, Johan Jeuring, Mihai Dascalu, Jordy van Dortmont, Gabriel Gutu-Robu, and Michiel Hulsbergen. Enhancing free-text interactions in a communication skills learning environment. In *Proceedings of the 13th international conference on Computer supported collaborative learning*, pages 363–364, June, 2019.
10. Jonathan Lessard. Designing natural-language game conversations. *Proc. DiGRA-FDG*, 2016.
11. Michael E Martinez. Cognition and the question of test item format. *Educational Psychologist*, 34(4):207–218, 1999.
12. Wookhee Min, Kyungjin Park, Joseph Wiggins, Bradford Mott, Eric Wiebe, Kristy Elizabeth Boyer, and James Lester. Predicting dialogue breakdown in conversational pedagogical agents with multimodal lstms. In *Proceedings of the Dialog System Technology Challenges Workshop (DSTC6)*, 2017.
13. Tom Murray. An overview of intelligent tutoring system authoring tools: Updated analysis of the state of the art. In *Authoring tools for advanced technology learning environments*, pages 491–544. Springer, 2003.
14. Olivia Realdon, Valentino Zurloni, Linda Confalonieri, Marcello Mortillaro, and Fabrizia Mantovani. Learning communication skills through computer-based interactive simulations. In *From communication to presence: Cognition, emotions and culture towards the ultimate communicative experience*, pages 281–303. IOS Press, Amsterdam, 2006.
15. Stefan Ruseti, Raja Lala, Gabriel Gutu-Robu, Mihai Dascalu, Johan Jeuring, and Marcell van Geest. Semantic matching of open texts to pre-scripted answers in dialogue-based learning. In *Proceedings of the 20th International Conference on Artificial Intelligence in Education*, volume 9387 of *LNCS*, pages 242–246, June, 2019.
16. Kurt Vanlehn. The behavior of tutoring systems. *International journal of artificial intelligence in education*, 16(3):227–265, 2006.