Authoring Conversational Intelligent Tutoring Systems

Zhiqiang Cai[1(✉)], Xiangen Hu[1,2], and Arthur C. Graesser[1]

[1] University of Memphis, Memphis, TN 38152, USA
{zcai, xhu, graesser}@memphis.edu
[2] Central China Normal University, Wuhan 430079, Hubei, China

Abstract. Conversational Intelligent Tutoring Systems (ITSs) are expensive to develop. While simple online courseware could be easily authored by teachers, the authoring of conversational ITSs usually involves a team of experts with different expertise, including domain experts, linguists, instruction designers, programmers, artists, computer scientists, etc. Reducing the authoring cost has long been a problem in the use of ITSs. Using AutoTutor as example, this paper discusses the authoring process and possible solutions by automatizing some authoring processes in authoring conversational ITSs.

Keywords: Conversational ITSs · AutoTutor · Authoring tools · Question generation · Short answer grading

1 Introduction

The first educational computer system, PLATO (Programmed Logic for Automatic Teaching Operation), was developed in 1960 [15]. In 1968, Bitzer and Shaperdes [2] claimed that large scale CBE (Computer-Based Education) systems could be built. As personal computer and internet came out, large scale courseware could be developed more and more easily. The best example is probably MOOCs (Massive Open Online Courses). In 2017, there were 81 million students registered to 9.4 thousand MOOCs in over 800 universities (https://www.class-central.com/report/mooc-stats-2017/).

The fast growth of MOOCs is due to the cheap and easy authoring, even though the effect is debatable [20, 27]. On the contrary, although ITSs have been proven to be as effective as human tutors [23, 24], they have never been so widely used as MOOCs. The reason behind this is simple: building ITSs is expensive. Let's have a look at the typical authoring process of a MOOC lesson.

A teacher had a powerpoint document for a lesson. The teacher presented the slides in a classroom and had the lecture video recorded. The teacher then uploaded the video to a MOOCs platform. Then the teacher announced that a MOOC course was up and students from all over the world could join the class online. Subsequent videos of later lectures were then continuously uploaded to the platform to complete the course. The teach might do a little more, such as uploading assignments, grading student homework, and answering student questions online. For a teacher who is giving a lecture to an offline class, such additional "authoring" processes are almost of zero cost. It is basically copying an offline class to an online platform. It is easy to convince schools to

R. A. Sottilare and J. Schwarz (Eds.): HCII 2019, LNCS 11597, pp. 593–603, 2019.
https://doi.org/10.1007/978-3-030-22341-0_46

use MOOCs because it could hugely expand class size with little investment. However, the major disadvantage of offline classroom teaching remains to exist - classroom teaching lacks adaption to individual student's needs.

Tutoring is an individualized learning process that cannot be pre-recorded. The interactions between a tutor and a student are generated on the fly. ITSs are created to simulate human tutors to help individual students, especially struggling students to improve their learning. Unlike common computed-based courseware that focuses on delivering knowledge, ITSs adaptively help individual student learn. An ITS system usually contains a domain model that represents domain knowledge by specific problems, a student model that tracks each student's learning history, a pedagogical model that utilizes successful pedagogical strategies, and an interface model that provides easy access to learning content, learning history and rich interactions [10]. While common courseware systems allow students to select one's own learning pace, ITSs go further – ITSs organize domain problems in a way that allows each individual student to go through an optimal learning path. A student model keeps track of complete learning history of each student. The learning data is then used for the pedagogical model to select learning path and provide adaptive feedback in each step. Authoring the content of an ITS system is usually a complicated process involving a team of experts with different expertise, including domain experts, language experts, computer scientist, instruction design expert, interface design experts, and more.

This paper describes AutoTutor authoring process as an example of conversational ITSs and explores possible ways that may automatize ITS authoring processes and thus reduce the cost of ITS authoring and increase the performance of the systems.

2 AutoTutor Systems

2.1 AutoTutor Conversation

AutoTutor is a system created by Arthur Graesser and his team in 1990s [13]. Since then, many AutoTutor applications have been successfully developed, including a Computer Literacy tutoring system [19, 25], a Newtonian Physics tutoring system [7], OperationARIES [16], ElectronixTutor [10], and many others. AutoTutor helps student learn by holding a conversation in natural language. AutoTutor is known for its Expectation-Misconception Tailored (EMT) conversation style [11]. An EMT conversation starts with a main question or a problem. The goal of the conversation is to help student construct a complete answer to the main question or a solution to the main problem. In each conversation turn, AutoTutor evaluates student input and finds the missing parts of the answer. AutoTutor then asks a hint or a prompt question targeting one of the missing parts. A student constructs a complete answer or solution by answering various hint or prompt questions. In AutoTutor, a hint question is a question that requires an answer with a sentence or a clause; while a prompt question requires an answer with a word or a phrase. When a misconception is identified, the misconception is corrected before a hint or prompt question is asked.

Figure 1 shows the flow of an EMT conversation. The conversation starts from the "Question". If the student's answer is complete, the conversation ends by presenting an ideal "Answer". If the student did not answer the question at all, the system goes into a "Pump" step, at which the tutor encourages the student to construction an answer. When a partial answer is received, the system enters an "Expectation/Misconception cycle". In this cycle, the tutor uses hints and prompts to help the student to construct answers that cover all expectations. When an expectation is covered, an "Assertion" is given. An "Assertion" is a statement about the expectation. After the student answers a prompt question, the tutor always gives the correct prompt completion answer ("PComp"). Misconception is identified in the hint or prompt answering process. When a misconception is identified, the tutor may immediately correct the misconception and continue the hint/prompt process.

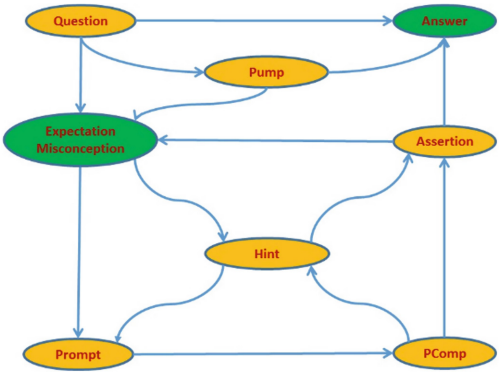

Fig. 1. Flowchart of AutoTutor EMT conversation.

The EMT conversation could be used in single agent systems, as well as multiple agent systems. When two agents are used, the conversation between a human student and two computer agents is called "trialog" [8]. In a trialog conversation, one compute agent usually plays a role as a tutor and the other agent plays a role as a peer student. With the help of the peer student, different conversation modes can be used. In the last decade, AutoTutor team have successfully developed four trialog modes, including vicarious learning mode, tutoring mode, teachable agent mode and competition mode. In the vicarious learning mode, the conversation is mostly between the teacher agent and the peer student agent. To keep the human student engaged, the human student is occasionally asked to answer some simple definitional or verificational questions during the conversation. This mode is used when the human student has very little knowledge about the topic under discussion. "Vicarious learning" refers to a learning strategy by which students learn by observing. This strategy has been proven effective [6, 9]. The tutoring mode conversation is mainly between the tutor agent and the human student. The peer student agent plays a role as a sidekick. The use of peer student in the tutoring mode is helpful especially when the answer from the human student is *likely* a wrong answer. So far there is no naturally language processing algorithm that can

identify a wrong answer with 100% accuracy. Giving a negative feedback to a correct answer due to a wrong judgement would be awkward. To avoid this, AutoTutor uses a safe conversation trick. When the system judges that the human student's answer is very likely wrong, the peer student chimes in and articulates a surely wrong answer which is semantically close to the one from the human student. Then the tutor agent critics the peer student agent without taking the risk of giving an awkward negative feedback to a correct answer. The tutoring mode is used when the human student has average knowledge about the topic. When the human student has high knowledge about the topic, the teachable agent could be used. In this mode, the conversation is mainly between the peer student and the human student. The peer student seeks help from the human student. The human student's responsibility is to teach the peer student until the peer student fully understand the topic. The tutor agent helps when the human student has problem in teaching. This provides the human student an opportunity to learn from teaching, which is also an effective learning strategy [14]. The competition mode is often used in a game like situation. The peer student and the human student compete in getting the answer to the main question or the solution to the main problem. This mode has been proven to be helpful in engaging human students [16].

EMT conversations are useful for constructing complex answers and solutions that involve deep understanding of the knowledge. To author an engaging lesson, other simple types of conversations could be used. For example, a greeting and introduction conversation could be used at the beginning of a lesson, a transition conversation could be used between two main questions, an instructional conversation could be used for a short quiz, a closing conversation could be used at the end of a lesson, etc.

2.2 AutoTutor Tutoring Interface

A typical AutoTutor interface contains five major components, including a main question area, an agent area, a media area, a conversation history area and an input box (see Fig. 2). The main question area displays a question under discussion. The main question usually stays there from the beginning to the end of a lesson. The agent area displays one or two animated talking heads. The talking heads deliver information by speech. The interactive media area displays texts, pictures, videos, and interactive controls, such as clickable buttons, dropdown menu items, drag-and-drop objects, text areas that can be highlighted, etc. The content displayed in the media area changes as the conversation goes. A student could interact with the controls in the media area and trigger events that may change the way how the conversation goes. The conversation history area displays the conversation between the agents and the human student. The student input box allows a student to enter utterances. Other components could be added. For example, a microphone component could be added to enable voice input, a camera component could be added to allow video input. Examples of such AutoTutor systems can be found at autotutor.org.

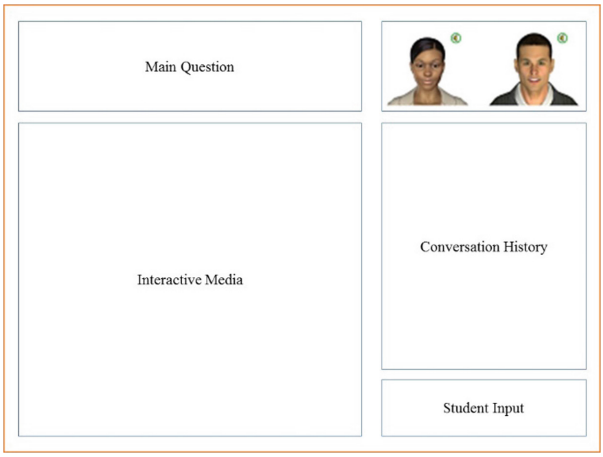

Fig. 2. AutoTutor Interface with five major components.

3 Authoring a Webpage-Based AutoTutor Lesson

Developing an AutoTutor application is a complicated process. The development tasks usually include creating conversational agents, developing interactive web pages, authoring conversation units and developing tutoring rules. None of these tasks could be easily done by a domain expert without expertise in web page development, conversation script authoring and intelligent tutoring. However, well designed authoring tools could make it possible for domain experts to put in most of the content and thus minimize the involvement of other developers. Pre-created computer agents, webpage templates and conversation unit template provided by AutoTutor authoring tools allow domain experts create agents, webpages and conversation units by replacing texts, images and videos in the templates. During the last decade, several ASATs (AutoTutor Script Authoring Tools) have been developed [3, 4]. Yet, improvements are still needed to make the tools easier for use.

3.1 Creating Agents

Creating agents is a process at application level. Once agents are created, they are used through all lessons of the application. An AutoTutor agent is usually an animated talking head that can deliver speeches and simple gestures. There are online character web services that can be easily integrated into AutoTutor systems. In addition to a name that matches the appearance and the voice of an agent, AutoTutor agents often store a list of commonly used expressions, such as greeting, agreeing, disagreeing, positive feedback, negative feedback, etc. Such expressions could be created to reflect an agent's "personality". For example, a negative feedback from a friendly agent could be "I don't think that is right"; while a straightforward agent may say "That is terribly wrong!" While more agents could be used, AutoTutor often uses one agent for dialog-based systems and two agents for trialog-based systems. To communicate with

AutoTutor interface, an agent component needs two basic functions. One is a *Speak* function that can receive a textual speech and deliver the speech in voice and gesture. The textual speech may contain Speech Synthesis Markup Language (SSML) tags to indicate gestures or special speech effects [22]. The other is a *Status Report* function to tell AutoTutor interface whether or not the agent is busy in speaking. Optionally, the agent may have an interruption function that allows the interface to pause or stop the speech when needed.

3.2 Developing Interactive Webpages

An interactive webpage is used to show a part of the problem under discussion or a step toward a solution. A webpage may contain static texts and images. It may also contain interactive elements, such as buttons, pulldown menus, highlightable texts, drag and drop objects, etc. Developing highly interactive webpages is often beyond what a domain expert can do. However, this task can be split into two different tasks. One task is for webpage developers to develop high quality interactive template webpages. The other task is for the domain experts to fill in the content in the template pages. The question is, can we provide enough templates so that domain experts can create satisfactory lessons? Well, we would say yes or no. We can think of common webpages that are used in learning environments, such as multiple-choice questions, highlightable texts, images with hotspots, and so on. Yet, there could always be some webpages a domain expert cannot find. Therefore, continuous template development and team authoring is probably unavoidable. On the other hand, even if webpage developers are actively available, template-based authoring is always more cost-effective.

An interaction on an AutoTutor webpage usually triggers an event, which could be associated with conversational units. For example, the selection of an answer to a multiple-choice question could trigger a "correct answer" event or "incorrect" answer event, which can be associated with corresponding feedback in a conversation. The event could also contribute to the assessment of student performance and make impact to the tutoring path. For example, the current webpage could be followed by a webpage with harder or easier items according to the performance of the human student.

3.3 Authoring Conversation Units

Conversation units could be of very different size. A small unit could be a feedback with a single utterance. A large one could be an EMT conversation that can selectively deliver tens or even hundreds of utterances. EMT is the typical and most complicated conversation unit. Other conversation types could be considered as simplified version of an EMT. In this section we focus on the authoring of an EMT unit.

Main Question. An EMT conversation starts with a main question. The length of the answer to the question determines the possible number of turns of the conversation. For a typical EMT conversation, the main question usually requires an answer of about three to ten sentences long. There are several issues that need to keep in mind when authoring an EMT main question. First, the question should be a deep reasoning question in order to support a deep comprehension conversation. The question schema can be found, for

example, in Graesser, Rus and Cai [12]. Second, the question should be of reasonable uncertainty. In other words, while there could be different ways to answer the main question, the answers should form a small number of semantically different clusters. A highly uncertain question could be, for example, "*What is your favorite work?*" There is no way to preset a possible ideal answer for this question because everyone may have a different answer. AutoTutor main questions are usually of low uncertainty. Here is an example: "*The sun exerts a gravitational force on the earth as the earth moves in its orbit around the sun. Does the earth pull equally on the sun? Explain why?*" More detailed discussions on question uncertainty can be found in Cai et al. [3].

Ideal Answer to Main Question. For a question with low uncertainty, usually a domain expert can form a reasonable ideal answer. For example, the ideal answer to the above sun and earth question could be:

> *The force of gravity between earth and sun is an interaction between these two bodies. According to Newton's third law of motion, if one body exerts a force on the other then the other body must exert an equal and opposite force on the first body. Therefore, the sun must experience a force of gravity due to the earth, which is equal in magnitude and opposite in direction to the force of gravity on the earth due to the sun.*

Expectations. After an ideal answer is authored, expectations can be formed by splitting the ideal answer to the main question. For example, the above ideal answer may be split into three expectations:

1. *The sun exerts a gravitational force on the earth.*
2. *Newton's third law says that for every action there is an equal and opposite reaction.*
3. *Thus, the force of the earth on the sun will be equal and opposite to the force of the sun on the earth.*

An expectation is a part of the ideal answer. It could a rewording of a sentence or a clause from the ideal answer. The collection of the expectations should cover all aspects of the ideal answer. The expectations may have overlap but should have enough difference in meaning so that they are semantically distinguishable.

Misconceptions. A misconception is a typical wrong answer based on incorrect thinking or misunderstanding. It is usually hard for authors to pre-imaging what misconception learners may have. Therefore, misconceptions are usually added when they are identified from learner's inputs. It is fine to keep misconception element blank in an initial EMT script.

Hints. A hint is a question to help the learner to construct one of the expectations. The answer to a hint should be a part of an expectation. Multiple hints could be created for each expectation. A hint should be asked in a way so that the learner would answer in a sentence. For example, "What is going on in this situation between the earth and sun?" The answer to this hint could be "The sun exerts a gravitational force on the earth." Since the purpose is to help the learner to construct the answer, a hint question should have minimum number of "juicy" answer words. For example, the above example hint question only used "sun" and earth and left "exerts" and "gravitational" for the learner to construct.

Prompts. In AutoTutor, a prompt is a question to help the learner to construct a small part (a word or a phrase) of an expectation. For example, "What is the type of force exerted by the sun on the earth?" The answer could be a single word, "gravity".

Hint and Prompt Answers. In addition to expert's answers to hint and prompt questions, AutoTutor requires authors to provide possible students answers. Such answers could be correct, partially correct, wrong, irrelevant or any other types. These answers are important because they are used for matching learners' inputs. In the initial authoring process, such answers are "imagined" by authors. Unfortunately, it is usually hard for an expert to exhaustively imagine all possible answers learners may give. In practice, new answers and answer types are added after learner answers are collected. Thus, AutoTutor EMT authoring is usually an iterative process.

Question Answering Pairs. AutoTutor style conservation is mostly "answer questioning", i.e., learners learn by answering computer agents' questions. However, AutoTutor also responds to learners' questions that are highly related to the topic. At the initial authoring phase, authors may prepare answers for questions that are likely asked by learners. Authors may also add answers to questions asked by learners when they interact with AutoTutor.

Agent Assignment. Questions and answers are assigned to specific agents, if there are multiple agents. In other words, a question could be asked by a tutor agent (e.g., in vicarious and tutoring mode) or a peer student agent (e.g., teachable agent mode). An answer could also be given by an agent. Tutor agent always gives a correct answer. Peer student may give any types of answers. The wording of a question or answer must be consistent with the role of the agent. For example, a tutor agent may say "The correct answer is gravity"; while a peer student agent may say "I think the answer is gravity".

3.4 Authoring Conversation Rules

Conversation rules are created to adaptively select displayable content, feedback and questions and change interface settings. Each AutoTutor lesson may have its own rule set. A rule set may be shared by multiple lessons. Each rule has two parts: condition part and action part. The condition part specifies the condition under which the rule could be selected; and the action part specifies a sequence of actions to execute when the rule is selected. The selection variables in the condition part include

(1) Status: representing the current conversation step;
(2) Response: the type of the learner's inputs (e.g., complete answer, partial answer, etc.);
(3) Event: representing learner's actions or interface changes, such as new content is loaded, a button is pressed, etc.;
(4) HasItem: a flag indicating whether a specific conversation item is available;
(5) Priority: selection priority when multiple rules satisfy (1) to (4);
(6) Frequency: a relative selection frequency used to compute random selection probability when multiple rules found according to (1) to (5).

The action part is a list of actions to be executed on the server side for further rule selection or on the interface side to display new content, deliver speeches or change interface settings. An action is an agent-act-data triplet. The agent could be "system", "tutor" or "peer student"; the act could be any action that can be executed by AutoTutor Conversation Engine (ACE) or by the interface program of the specific AutoTutor application; the data is additional information needed to execute the action.

We cannot cover more details about the rules due to the space limitation. Readers who are interested in knowing more about AutoTutor rules may find an example AutoTutor script template at http://ace.autotutor.org/attrialogtemplate001.xml.

4 Automatization in AutoTutor Script Authoring

High quality conversation script authoring is a difficult task. Just imagine writing conversation scripts for a Hollywood movie. A writer may write the conversation for each character in the story scene by scene linearly. However, in writing conversation scripts for ITS, there is an unknown character – the learner. An author may easily determine what the tutor, the peer student or any other computer agent should say without interacting with the learner. However, once the learner is considered, the authoring becomes complicated. In each conversation turn, an author needs to imagine all possible ways a learner may say and prepare for the next turn agent scripts based on each type of learner responses. Assuming there is a single script *S-1* for the first conversation turn. For the second turn, because AutoTutor needs to continue the conversation adapting to the learner's input, there will be multiple scripts *S-2-1, S-2-2, ..., S-2-N*. The number of scripts may exponentially increase with the number of turns.

AutoTutor solved this problem by restricting the conversation within the EMT framework, so that the conversation paths always converge to the answer of the main question. Authoring AutoTutor script becomes easier with the help of AutoTutor Script Authoring Tools. However, understanding and authoring high quality conversation units is still challenging to domain experts. The recent advances in natural language processing (NLP) [21] may help to make AutoTutor script authoring easier.

Question generation techniques may be used in AutoTutor hint and prompt generation [5, 18]. AutoTutor targets deep knowledge and requires deep reasoning questions to achieve the goal [12]. Using automatic question generation techniques, authoring load is reduced from creating questions to selecting and editing questions. Moreover, the automatically generated deep questions could be good examples to help authors construct high quality questions.

Automatic question answering techniques may be used in AutoTutor so that authors don't have to prepare answers for possible student questions. Currently, question answering systems are mostly information retrieval systems [17]. Conversation systems like AutoTutor requires answers in the style of "conversation". A conversational answer needs to be adequately conveyed through speeches. Such answers are usually short and clear. Long answers could be delivered as conversation between multiple computer agents and the learner. Therefore, a conversational question answering involves two types of language processing: (1) generating an answer to a given question; and (2) converting an answer to conversational scripts.

Text clustering techniques could be very helpful in data-driven script authoring. Learners' answers could be collected and presented as semantic clusters so that authors can create adaptive feedback [1, 26]. Iterative answer clustering could increase the accuracy of user input assessment and thus increase system performance.

5 Conclusion

AutoTutor EMT framework successfully solved the conversation turn explosion problem. Yet, authoring effective conversational ITS like AutoTutor is still an expensive process. Using automatic natural language techniques may help reduce the cost and increase the quality of authoring. Would it be possible in the future that effective conversational tutoring scripts can be fully automatically generated? That may happen at a time after Hollywood movie scripts could be fully automatically generated.

Acknowledgment. The research on was supported by the National Science Foundation (DRK-12-0918409, DRK-12 1418288), the Institute of Education Sciences (R305C120001), Army Research Lab (W911INF-12-2-0030), and the Office of Naval Research (N00014-12-C-0643; N00014-16-C-3027). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF, IES, or DoD. The Tutoring Research Group (TRG) is an interdisciplinary research team comprised of researchers from psychology, computer science, and other departments at University of Memphis (visit http://www.autotutor.org).

References

1. Aggarwal, C.C., Zhai, C.: A survey of text clustering algorithms. In: Aggarwal, C., Zhai, C. (eds.) Mining Text Data. Springer, Boston (2012). https://doi.org/10.1007/978-1-4614-3223-4_4
2. Bitzer, D.L., Skaperdas, D.: PLATO IV-An Economically Viable Large Scale Computer-Based Education System. National Electronics (1968)
3. Cai, Z., Gong, Y., Qiu, Q., Hu, X., Graesser, A.: Making AutoTutor agents smarter: AutoTutor answer clustering and iterative script authoring. In: Traum, D., Swartout, W., Khooshabeh, P., Kopp, S., Scherer, S., Leuski, A. (eds.) IVA 2016. LNCS (LNAI), vol. 10011, pp. 438–441. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47665-0_50
4. Cai, Z., Graesser, A.C., Hu, X.: ASAT: AutoTutor script authoring tool. In: Sottilare, R., Graesser, A.C., Hu, X., Brawner, K. (eds.) Design Recommendations for Intelligent Tutoring Systems: Authoring Tools, pp. 199–210. Army Research Laboratory (2015)
5. Cai, Z., Rus, V., Kim, H.-J.J., Susarla, S.C., Karnam, P., Graesser, A.C.: NLGML: a Markup Language for question generation. In: Proceedings of E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education, Honolulu, Hawaii, USA, pp. 2747–2752, October 2006
6. Craig, S.D., Gholson, B., Ventura, M., Graesser, A.C.: Overhearing dialogues and monologues in virtual tutoring sessions: effects on questioning and vicarious learning. Int. J. Artif. Intell. Educ. **11**, 242–253 (2000)

7. Graesser, A., et al.: Why/AutoTutor: a test of learning gains from a physics tutor with natural language dialog AutoTutor and why/AutoTutor. In: Proceedings of the 25th Annual Conference of the Cognitive Science Society, Boston, pp. 474–479 (2003)
8. Graesser, A.C.: Conversations with AutoTutor help students learn. Int. J. Artif. Intell. Educ. **26**(1), 124–132 (2016)
9. Graesser, A.C., Forsyth, C.M., Foltz, P.: Assessing conversation quality, reasoning, and problem-solving performance with computer agents. In: The Nature of Problem Solving: Using Research to Inspire 21st Century Learning (2017)
10. Graesser, A.C., Hu, X., Sottilare, R.: Intelligent tutoring systems. In: International Handbook of the Learning Sciences (2018)
11. Graesser, A.C., et al.: AutoTutor: a tutor with dialogue in natural language. Behav. Res. Methods Instrum. Comput. **36**(2), 180–192 (2004)
12. Graesser, A.C., Rus, V., Cai, Z.: Question Classification Schemes. In: The Workshop on Question Generation (2008)
13. Graesser, A.C., Wiemer-Hastings, K., Wiemer-Hastings, P., Kreuz, R.: AutoTutor: a simulation of a human tutor. Cogn. Syst. Res. **1**(1), 35–51 (1999)
14. Leelawong, K., et al.: Teachable agents learning by teaching environments for science domains. In: Proceedings of the Fifteenth Annual Conference on Innovative Applications of Artificial Intelligence, pp. 109–116 (2003)
15. Van Meer, E.: PLATO: From computer-based education to corporate social responsibility. Iterations Interdiscip. J. Softw. Hist. **2**, 1–22 (2003)
16. Millis, K., Forsyth, C., Butler, H., Wallace, P., Graesser, A., Halpern, D.: Operation ARIES!: a serious game for teaching scientific inquiry. In: Ma, M., Oikonomou, A., Jain, L.C. (eds.) Serious Games and Edutainment Applications, pp. 169–195. Springer, London (2011). https://doi.org/10.1007/978-1-4471-2161-9_10
17. Mishra, A., Jain, S.K.: A survey on question answering systems with classification. J. King Saud Univ. Comput. Inf. Sci. **28**(3), 345–361 (2016)
18. Olney, A.M., Graesser, A.C., Person, N.K.: Question generation from concept maps. Dialogue Discourse **3**(2), 75–99 (2012). Editors: Paul Piwek and Kristy Elizabeth Boyer
19. Person, N.K., Graesser, A.C., Kreuz, R.J.: Simulating human tutor dialog moves in AutoTutor (2001)
20. Rezaei, E., Zaraii Zavaraki, E., Hatami, J., Abadi, K.A., Delavar, A.: The effect of MOOCs instructional design model-based on students' learning and motivation. Man India **97**(2017), 115–126 (2017)
21. Sun, S., Luo, C., Chen, J.: A review of natural language processing techniques for opinion mining systems. Inf. Fusion **36**, 10–25 (2017)
22. Taylor, P., Isard, A.: SSML: a speech synthesis Markup Language. Speech Commun. **21**(1–2), 123–133 (1997)
23. Vanlehn, K.: The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. Educ. Psychol. **46**(4), 197–221 (2011)
24. Vanlehn, K., Graesser, A.C., Jackson, G.T., Jordan, P., Olney, A., Rosé, C.P.: When are tutorial dialogues more effective than reading? Cogn. Sci. **31**(1), 3–62 (2007)
25. Wiemer-Hastings, P., Graesser, A.C., Harter, D., Grp, T.R.: The foundations and architecture of autotutor. Intell. Tutoring Syst. **1452**, 334–343 (1998)
26. Xu, J., et al.: Self-Taught convolutional neural networks for short text clustering. Neural Netw. **88**, 22–31 (2017)
27. Yousef, A.M.F., Chatti, M.A., Schroeder, U., Wosnitza, M., Jakobs, H.: A review of the state-of-the-art. In: Proceedings of the 6th International Conference on Computer Supported Education - CSEDU 2014, pp. 9–20 (2014)