AIED 2009: 14th International Conference on Artificial Intelligence in Education

Workshops Proceedings

Editors and Co-Chairs:

Scotty D. Craig
University of Memphis, USA

Darina Dicheva
Winston-Salem State University, USA

July 6-7th, 2009
Brighton, UK

Preface

The supplementary proceedings of the workshops held in conjunction with AIED 2009, the fourteen International Conference on Artificial Intelligence in Education, July 6-7, 2009, Brighton, UK, are organized as a set of volumes - a separate one for each workshop.

The set contains the proceedings of the following workshops:

- **Volume 1: The 2nd Workshop on Question Generation**
  Co-chairs: Vasile Rus & James Lester. *University of Memphis, USA & North Carolina State University, USA.*
  http://www.questiongeneration.org/AIED2009/

- **Volume 2: SWEL'09: Ontologies and Social Semantic Web for Intelligent Educational Systems**
  Co-chairs: Niels Pinkwart, Darina Dicheva & Riichiro Mizoguchi. *Clausthal University of Technology, Germany; Winston-Salem State University, USA & University of Osaka, Japan.*
  http://compsci.wssu.edu/iis/swel/SWEL09/index.html

- **Volume 3: Intelligent Educational Games**
  Co-chairs: H. Chad Lane, Amy Ogan & Valerie Shute. *University of Southern California, USA; Carnegie Mellon University, USA & Florida State University, USA.*
  http://projects.ict.usc.edu/aied09-edgames/

- **Volume 4: Scalability Issues in AIED**
  Co-chairs: Lewis Johnson & Kurt VanLehn. *Alelo, Inc., USA & Arizona State University, USA.*
  http://alelo.com/aied2009/workshop.html

- **Volume 5: Closing the Affective Loop in Intelligent Learning Environments**
  Co-chairs: Cristina Conati & Antonija Mitrovic. *University of British Columbia, Canada & University of Canterbury, New Zealand.*
  http://aspire.cosc.canterbury.ac.nz/AffectLoop.html

- **Volume 6: Second Workshop on Culturally-Aware Tutoring Systems (CATS2009): Socio-Cultural Issues in Artificial Intelligence in Education**
  Co-chairs: Emmanuel G. Blanchard, H. Chad Lane & Danièle Allard. *McGill University, Canada; University of Southern California, USA & Dalhousie University, Canada.*
  http://www.iro.umontreal.ca/~blanchae/CATS2009/

- **Volume 7: Enabling Creative Learning Design: How HCI, User Modelling and Human Factors Help**
  Co-chairs: George Magoulas, Diana Laurillard, Kyparisia Papanikolaou & Maria Grigoriadou. *Birkbeck College, University of London, UK; Institute of Education, UK; School of Pedagogical and Technological Education, Athens, Greece & University of Athens, Greece*.
  https://sites.google.com/a/lkl.ac.uk/learning-design-workshop/Home

- **Volume 8: Towards User Modeling and Adaptive Systems for All (TUMAS-A 2009): Modeling and Evaluation of Accessible Intelligent Learning Systems**
  Co-chairs: Jesus G. Boticario, Olga C. Santos and Jorge Couchet, Ramon Fabregat, Silvia Baldiris & German Moreno. *Spanish National University for Distance Education, Spain & Universitat de Girona, Spain*.
  https://adenu.ia.uned.es/web/es/projects/tumas-a/2009

- **Volume 9: Intelligent Support for Exploratory Environments (ISEE'09)**
  Co-chairs: Manolis Mavrikis, Sergio Gutierrez-Santos & Paul Mulholland. *London Knowledge Lab, Institute of Education/Birkbeck College, University of London, UK & Knowledge Media Institute and Centre for Research in Computing, Open University, UK*.
  http://link.lkl.ac.uk/isee-aied09

- **Volume 10: Natural Language Processing in Support of Learning: Metrics, Feedback and Connectivity**
  Co-chairs: Philippe Dessus, Stefan Trausan-Matu, Peter van Rosmalen & Fridolin Wild. *Grenoble University, France; Politehnica University of Bucharest; Open University of the Netherlands & Vienna University of Economics and Business Administration, Austria*.
  http://webu2.upmf-grenoble.fr/sciedu/nlpsl/

While the main conference program presents an overview of the latest mature work in the field, the AIED2009 workshops are designed to provide an opportunity for in-depth discussion of current and emerging topics of interest to the AIED community. The workshops are intended to provide an informal interactive setting for participants to address current technical and research issues related to the area of Artificial Intelligence in Education and to present, discuss, and explore their new ideas and work in progress.

All workshop papers have been reviewed by committees of leading international researchers. We would like to thank each of the workshop organizers, including the program committees and additional reviewers for their efforts in the preparation and organization of the workshops.

July, 2009
Scotty D. Craig and Darina Dicheva

AIED 2009 Workshops Proceedings
Volume 1

The 2nd Workshop on Question Generation

Workshop Co-Chairs:

Vasile Rus
University of Memphis, USA

James Lester
North Carolina State University, USA

http://www.questiongeneration.org/AIED2009/

Preface

Question asking has frequently been considered a fundamental cognitive process in the fields of education and cognitive science. The ideal learner is an active, self-motivated, creative, inquisitive person who asks deep questions and searches for answers to such thought-provoking questions. Real learners on the other hand are less inquisitive, and thus modern learning environments aim at modeling and scaffolding question asking as a way to boost learning gains in students.

Question Generation is an important component in learning technologies such as Intelligent Tutoring Systems, inquiry-based environments, and instructional games. Advances in Question Generation will have ripple effects in learning technologies that rely on Question Generation to provide quality education to students of various ages and disciplines.

Question Asking/Generation can be introduced into various learning scenarios (e.g., dialogue-based or vicarious learning) through a Question Generation component. The Question Generation component can be semi-automated (e.g., it helps subject matter experts generate questions as part of the authoring tools used to create content in the form of curriculum scripts) or fully automated (currently, this is the case for simple types of questions such as multiple-choice questions).

This workshop is the second in a series of workshops that began with the NSF Workshop on the Question Generation Shared Task and Evaluation Challenge (www.questiongeneration.org) held in September 2008 in Arlington, Virginia, USA. It solicits the involvement of participants across disciplines ranging from Artificial Intelligence in Education and Psycholinguistics/Discourse Processes to Natural Language Generation on cognitive and computational aspects of question generation.

The workshop is being organized as a 1-day workshop. There are two sets of sessions. In the morning sessions, regular paper presentations on general topics related to QG will be scheduled. The afternoon sessions will be dedicated to discussions and presentations related to QG in Intelligent Tutoring Systems, one category of shared tasks identified at the previous *Workshop on The Question Generation Shared Task and Evaluation Challenge*. As part of the afternoon sessions we will have a student session.

The workshop was supported by the National Science Foundation through a grant (RI-0938239) that provided travel support to students attending the workshop.

We would like to thank our outstanding Program Committee members for their help with making the workshop a resounding success. Also, we would like to thank Dr. Tanya Korelsky from the National Science Foundation for her continuing support of the young and fast-growing Question Generation research community.

July, 2009
Vasile Rus and James Lester

Program Committee

Co-Chair: Vasile Rus, *University of Memphis, USA* (vrus@memphis.edu)
Co-Chair: James Lester, *North Carolina State University, USA* (lester@csc.ncsu.edu)

Delphine Bernhard, *Darmstadt Technical University, Germany*
Kristy Elizabeth Boyer, *North Carolina State University, USA*
Yllias Chali, *University of Lethbridge, Canada*
Dan Flickinger, *Stanford University, USA*
Corina Forascu, *Al. I. Cuza University, Romania*
Donna Gates, *Carnegie Mellon, USA*
Natália Giordani, *Federal University of Rio de Janeiro, Brasil*
Art Graesser, *University of Memphis, USA*
Michael Heilman, *Carnegie Mellon, USA*
Aravind Joshi, *University of Pennsylvania, USA*
Chin-Yew Lin, *Microsoft Research Asia, China*
Mihai Lintean, *University of Memphis, USA*
Tomasz Marciniak, *Yahoo Research, UK*
Ruslan Mitkov, *University of Wolverhampton, UK*
Jack Mostow, *Carnegie Mellon, USA*
Rodney D. Nielsen, *University of Colorado, USA*
Jose Otero, *University of Alcala, Spain*
Juan Pino, *Carnegie Mellon, USA*
Paul Piwek, *The Open University, UK*
Rashmi Prasad, *University of Pennsylvania, USA*
Lucy Vanderwende, *Microsoft, USA*

Additional Reviewers

Wei Chen, *Carnegie Mellon, USA*

Table of Contents

What a Pilot Study Say About Running a Question Generation Challenge 1
Lee Becker, Rodney Nielsen and Wayne Ward

An Empirically Derived Question Taxonomy for Task Oriented Tutorial Dialogue 9
Kristy Elizabeth Boyer, William Lahti, Rober Phillips, Michael Wallis, Mladen Vouk, and James Lester

Generating Questions Automatically from Informational Text 17
Wei Chen, Gregory Aist, and Jack Mostow

Question Generation: Taxonomies and Data 25
Corina Forăscu and Iuliana Drăghici

Ranking Automatically Generated Questions as a Shared Task 30
Michael Heilman and Noah Smith

Generation of Exercises within the PERLEA project 38
Stéphanie Jean-Daubias, Marie Lefevre, and Nathalie Guin

AMBRE-teacher: a Module Helping Teacher to Generate Problems 43
Stéphanie Jean-Daubia and Nathalie Guin

Building Resources for an Open-Task on Question Generation 48
Vasile Rus, Eric Woolley, Mihai Lintean, and Arthur Graesser

Influence of Reading Goals on Question Generation 53
Vincente SanJosé, Koto Ishiwa, and José Otero

Increasing Problem Simplification as Scaffolding in Exercises 58
Tsukasa Hirashima

Generating Questions from OpenLearn study units 66
Brendan Wyse and Paul Piwek

What a pilot study says about running a question generation challenge

Lee BECKER [a], Rodney D. NIELSEN [a,b] and Wayne H. WARD [a,b]
[a] *The Center for Computational Language and Education Research, University of Colorado at Boulder*
[b] *Boulder Language Technologies*

Abstract.
We present a pilot study, wherein annotators rated the quality of questions produced by our system and human tutors. This study has helped us to evaluate our tutorial dialog system's question generation abilities and has aided us in identifying areas for further refinement of our dialog strategy. Moreover, this experiment has taught us several important lessons and highlighted critical issues related to running a question generation challenge.

Keywords. Question Generation, dialog, intelligent tutoring systems

Introduction

In interactive educational technologies and intelligent tutoring systems the task of question generation (QG) encompasses more than deriving a surface form from a concept; this task includes choosing the appropriate question for the situation. Ideal questions in this domain not only address learning goals and learner knowledge gaps, they exhibit pertinence by grounding themselves in the context of the conversation while simultaneously maximizing learning gains.

Similarly, Vanderwende [7] argues that choosing which question to generate is as important as generating the question itself. She proposes that meaningful evaluation of question quality should focus on judging the *importance* of a question with respect to a larger text. Furthermore, Nielsen [4] states that identifying key concepts is a critical subtask in the QG process.

This problem of choosing an important question bears much similarity to the task of dialog act selection in spoken dialog systems wherein a dialog manager must produce follow-up utterances that are on-topic and aligned with user and system goals. For this paper we approach QG as a dialog management task, and present an experiment in which we assess our tutorial dialog system's question generating capabilities relative to human tutors. In the following sections we describe our motivations, give a brief overview of our dialog-based QG system, detail our question evaluation experiment, present and discuss our results, expound on the difficulties in defining a QG challenge, and close with suggestions for future work.

1. Motivations and Background

Our project's overarching goal is to develop a dialog based tutoring system for elementary school aged students. Our curriculum is based on the Full Option Science System (FOSS) [5] a proven research-based science curriculum system that has been employed in American schools for over a decade. FOSS consists of sixteen diverse science teaching and learning modules covering life science, physical science, earth and space science, scientific reasoning, and technology. For this study, we limit our coverage of FOSS to investigations about magnetism and electricity.

The system's dialog strategies are informed by the Questioning the Author (QtA) [1] teaching method. QtA is an approach to classroom instruction that uses dialog interaction to facilitate development of effective comprehension strategies and deep learning. When applying QtA to FOSS, instructors ask open-ended questions leading to dialogs that encourage the student to make sense of their hands-on experiences. Example questions include "What's going on here?", "What's that all about?", or "How does this connect to. . . ?"

2. System Overview

To generate tutorial dialog moves, we use a modified version of Phoenix [3], a frame and slot-based dialog management system. Target concepts from the lesson are modeled as propositions using the Phoenix frame representation, while sub-concepts are expressed as elements within the frame. During the course of conversation, our system picks manually-authored questions from a pool of questions associated with the frame currently in focus.

Figure 1 shows a simplified frame, its elements, and the pool of questions used to converse about the concept *Electricity flows from negative to positive*. If a concept is only partially addressed by the student, the system will take the next move from the pool of questions corresponding to an empty slot. Additionally, our version of Phoenix has a rule mechanism [2], which allows the system to act based on the content of the elements, and is useful for detecting and addressing misconceptions. The rule shown in Figure 1 triggers when a student is confused about the direction of current in a circuit.

Currently, the general strategy for authoring these frames is to start with an open-ended QtA style question and gradually increase question specificity until the targeted slot is filled. It should be noted that frame actions are multimodal and can include visuals and other multimedia content in addition to spoken dialog.

Though our questions are manually-authored, our system does attempt to choose pertinent or important questions given the conversational context. More importantly the system provides a framework for experimenting with various tutoring styles and questioning strategies.

3. Experiment

Our study was primarily motivated by a desire to see how well questions suggested by our dialog management system compared to questions generated by humans, but it also presented an opportunity to investigate the issues involved in running a QG challenge.

```
Frame: FlowInSeries
[_start]+
    Action: "We've been talking about components in a series circuit
             sharing a single path to a power energy source.
             Tell me about what's going on here"
[Electricity]
[Flow]+
    Action: "What are the blue dots showing?"
    Action: "How can you connect what you notice with the blue dots to
             your thinking about the flow of electricity?"
    Action: "Let's look at this again.  What is going on here?"
[FromTerminal]+
    Action: "You mentioned something about the flow of electricity.
             Tell me more about what is going on with that in this picture"
    Action: "How does the electricity flow in this circuit?"
    Action: "Tell me about how the electricity flows in and out of
             the battery"
[ToTerminal]+
    Action: "Tell me more about where the electricity is flowing to."
    Action: "What side of the battery is the electricity flowing towards?"
[_done]+
    Action: "You've made some good observations.
             Now let's talk more about..."
Rules:
[FromTerminal] ==  "Positive" OR [ToTerminal] == "Negative"
    Action: "Look closely at the flow of electricity.
             Can you tell me again about which way the
             electricity is flowing?"
```

Figure 1. Example Phoenix frame for proposition: *Electricity flows from negative to positive*

Like the Bystander Turing Test conducted by Person and Graesser [6], our study uses snippets of tutorial dialog to provide context for generating questions and then later evaluating them. Unlike the the tests carried out by Person and Graesser, our judges were not asked to ascertain whether the question was generated by a human or computer. Instead our instructions simply asked the evaluator to give a single score to the question taking into account factors like appropriateness to the context and felicity to QtA practices.

3.1. Participants

The six participants in this study are tutors employed by Boulder Language Technologies. Of the six tutors, three are considered experts in Questioning the Author (QtA) style tutoring and in the FOSS curriculum, while the other three have been given training in QtA and FOSS but are not considered to be at an expert level of proficiency. Five of the six participants assisted in producing questions specifically for this experiment, while all six took part in the evaluation of questions. Tutorial sessions were conducted with students in grades 3-6.

3.2. Question Generation / Collection

Transcripts used in this experiment were collected from computer-mediated one-on-one human tutoring sessions. In this environment, the student interacts with a virtual agent

Student Utterances:

- good
- <um> studying magnetism
- <um> there's a scale and some cups and washers p- <um> <side_speech> <uh> magnets and i forget the other thing the yellow things <um> <breath> <um> <fp> you would put the cup in in the scale and then you would put the
- the magnet post <um> on the s- under the cup <side_speech> you put a ma- a magnet in the cup and then you would put the other cup on the left st- side and you'd try and see how many washers you could get in that other cup
- the washers <breath> you when you you aren't gonna try and see how many you can fit in without <um> the force of the magnets breaking <um> since the washers are steel
- <um> yes ((i say)) so go in on the right side of the cup <breath> a- and you put them in softly <breath> then you can fit more

Expert Tutor Question:	what is important about putting the washers in softly?
Non-Expert Tutor Question:	tell me about the spacers
System Question:	what's going on between the two magnets?

Figure 2. Example dialog context (student utterances) and corresponding generated questions

that talks and presents visual materials. Behind the scenes, a human operator controls the agent, deciding which questions to ask and which visuals to display similar to a Wizard-of-Oz experiment. No deception was used; students were told that an experimenter would be available to help the agent. These sessions were conducted by a total of 8 different tutors (2 expert, 6 non-expert).

From these transcripts we randomly sampled 50 expert tutor dialog turns and 50 non-expert turns for evaluation. We then fed the corresponding student dialog, from its beginning to the sampled point, turn-by-turn into our Phoenix system and included the last question generated in our evaluation. Similarly, we generated a third question for evaluation by presenting the same student dialog, approximately one turn every three seconds, to our human tutors, requesting a tutor turn at the sampled point. See section 6 for issues that led to this methodology.

This process yielded 3 questions for each of 100 dialog contexts: an expert tutor question, a non-expert tutor question, and a Phoenix system question – creating a total of 300 questions for evaluation, with half of the expert and non-expert questions being pulled from the actual transcripts and the other half being generated as part of the experiment. An example dialog context and its associated questions are shown in Figure 2.

3.3. Question Evaluation

Each of the 300 questions were evaluated by both an expert tutor and a non-expert tutor. Special care was taken to ensure that evaluators never rated a question for a context where they themselves generated a question. The evaluation environment was similar to the question collection environment, wherein the participant was shown a sequence of student utterances turn-by-turn. After reading the context, a follow-up question was shown, and the participant was asked to give a single score on a scale of 1 (worst) to 5 (best) taking into account factors like how well it followed the context and how well it adhered to QtA principles.

Table 1. Independent Samples t-tests for ratings by evaluator and question generator. Scores were normalized for differences in scoring among individual evaluators, such that scores for a given evaluator had mean = 3.223 (the mean of all evaluations for all questions) and standard deviation = 1.0.

Evaluators	Phoenix v. Expert		Phoenix v. Non-Expert		Expert v. Non-Expert	
All Tutors	t=3.365	p=0.000	t=5.521	p=0.000	t=0.471	p=0.638
Exp. Tutors	t=5.021	p=0.000	t=4.665	p=0.000	t=0.340	p=0.734
Non-Exp. Tutors	t=1.999	p=0.047	t=3.152	p=0.002	t=0.984	p=0.326

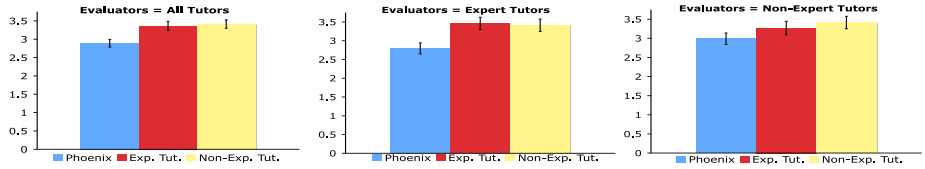

Figure 3. Comparisons of normalized mean scores and 95% confidence intervals by evaluator type.

Table 2. Inter-rater Group Correlations. The first column lists the questions' generator, while the other columns shows correlations over questions rated by evaluators from different classes. In columns where the groups are identical (i.e. expert v. expert), the correlations are computed over a subset of questions that were rated in two passes, otherwise the correlations are computed over all questions.

Question Generator	Expert v. Expert	Non-Expert v. Non-Expert	Expert v. Non-Expert
Phoenix	0.490 (p=0.006)	0.311 (p=0.095)	0.341 (p=0.001)
Expert Tutors	0.418 (p=0.021)	0.497 (p=0.007)	0.440 (p=0.000)
Non-Expert Tutors	0.458 (p=0.011)	0.650 (p=0.002)	0.500 (p=0.000)

4. Results

To fairly compare scores between evaluators, question scores were normalized for differences among individual evaluators, so that the mean score for each evaluator was the average score of all evaluations for all questions ($\mu = 3.223$) and the standard deviation was 1.

A series of independent samples t-tests were performed (Table 1 and Figure 4) and found that questions from both classes of tutors significantly outscored the Phoenix system. The difference in ratings of expert-generated questions and non-expert-generated questions was not statistically significant, holding true for both expert evaluators and non-expert evaluators, though one annotator on average gave higher ratings to the Phoenix questions than expert questions.

To get a sense of inter-rater reliability, 90 of the questions were scored in a second pass by another pair of tutors. These were used to compute Spearman rank-order correlation coefficients (Table 2) between tutors of the same group (Expert v. Expert, Non-Expert vs. Non-Expert). The rating from the original 300 evaluations were used to compute correlations across groups (Expert v. Non-Expert). There was positive correlation between all combinations on all groups of questions.

Lastly, Table 3 illustrates the difference in perceived quality between transcript derived human tutor questions, the experimentally generated human tutor questions, and Phoenix generated questions. Independent sample t-tests (Table 3) found significant differences between ratings for experimentally generated questions by human tutors and for questions generated via the other two approaches.

Table 3. Independent Sample t-tests between ratings for questions generated under different conditions (experimental human tutor, transcript derived, and phoenix generated).

Evaluators	Q. Gen. Cond 1	Mean Score 1	Q. Gen. Cond 2	Mean Score 2	t	p-value
All Tut.	Experimental	3.715	Transcript	3.064	6.810	0.000
	Experimental	3.715	Phoenix	2.892	9.062	0.000
	Transcript	3.064	Phoenix	2.892	1.816	0.070

5. Discussion

The most significant finding was the difference in perceived quality between questions extracted from transcripts and questions generated during the experiment. There are at least four factors that could contribute to this difference. First, tutor question asking abilities may have improved since the time the tutorial sessions in the transcripts were conducted. Additionally tutors may not have felt time constrained when writing questions during the experiment like they would during a live tutoring session. Third, observer effects might have played a role, as tutors knew the questions were going to be evaluated. Lastly the evaluation context may be a factor.

In our experiment the evaluation condition perfectly matched the condition for generating the experimental human tutoring question – they each viewed the student dialog turn-by-turn without generating or viewing the tutoring dialog that elicited it, whereas the Phoenix and transcript-derived questions relied on the additional context of their own prior turns. If Phoenix or the transcript tutor already asked a question very similar to the higher-rated experimentally-derived human tutor question, they would likely ask a more specific question, which would be perceived as being of lower quality. As discussed in the next section, this confound could not easily be avoided in the present experiment and has significant implications for a QG challenge task.

Sparse grammar coverage was also a significant factor contributing to the large divergence in ratings between human tutor generated questions and system-generated questions. Without the proper mapping of student utterances to semantic frames, the system is unable to align to the appropriate context, and consequently, is less able to ask a pertinent question.

A confounding factor in this experiment is our inability to synchronize the system state with the focus of the human tutor over the time course of the dialog. Though it is a goal to have the system's state closely match the human tutor's state, there is no way to ensure this. To determine how divergence between system and tutor state could affect question ratings, we analyzed scores by number of student turns taken before arriving at the question of interest; however we found only a minor downward trend in the ratings of system-generated data as the context grew.

6. Issues in Running a Question Generation Challenge

The results above suggest that scores were highest when the evaluation conditions better aligned with the conditions under which the question was generated, indicating that extra effort must be taken to ensure fair evaluation when running a QG challenge. In designing this experiment we debated the merits and drawbacks of several options concerning the dialog context and initialization of our system. The main approaches we considered were as follows:

1. Original transcript dialog context
2. Mixed transcript dialog context
3. System only transcript dialog context
4. Manually initialize system state
5. Combine all student turns
6. Student turns only

In the *original transcript* approach the dialog context presented during QG and evaluation would have both the human tutor and student turns from the original tutoring session. We decided not to use this approach because there would be no way for the system to recognize, model and/or factor tutor turns into the QG process.

The *mixed transcript* approach would present the dialog context as an interleaving of student turns from transcripts with system-generated turns/questions. With this approach student turns would not be in dialog alignment with the system-generated questions and could potentially lead to unnatural and confusing dialogs.

The *system only transcript dialog context* approach follows the approach used in a bystander Turing test, where the transcript utilized is from a pure system tutorial session. This approach would have been the ideal way of evaluating our system, but we could not carry out this experiment because we have yet to collect tutorial session transcripts since incorporating intelligence into our system. This issue presents great difficulty for a QG challenge, where many systems are being evaluated and there is no single system that can provide the dialog context leading back to the problems associated with options one and two above.

With the *manually initialize system state* approach the system's internal state is manually set to reflect gold standard student understanding and dialog context. Evaluation and QG would be carried out using the full dialog context from the original transcript. While this context would have provided a meaningful way to isolate and evaluate the system's question selection capabilities, we did not have the time or infrastructure to carry out an experiment centered around this approach. Requiring knowledge like this in a QG challenge would lead to significant manual intervention and additional system building by participants and would likely invalidate many of the results.

Instead of inputting student utterances turn-by-turn the *combine all student turns* approach would concatenate all the student utterances and feed them into the system as a single turn. The major drawback to this method is the lack of contextual constraint. A combined utterance may cover several concepts meaning there is no single appropriate concept upon which to focus; presumably most of the prior dialog revolved around key concepts.

As stated before, we opted to forego the first five alternatives and use a dialogue context consisting of *student turns only* to avoid many of the issues associated with the other approaches, however in doing so we may have introduced the generating condition biases discussed in section 5.

These confounding factors and our results demonstrate the difficulty in defining a dialog-based based QG task. Systems that make question generating decisions based on their own previous questions will be at a disadvantage for such an evaluation. We believe that requiring systems to add logic to account for input from an arbitrary external QG source would add a significant barrier to participation in a challenge, requiring system logic that, for most, would never be used in an end-user application. Perhaps the only way to fairly evaluate such systems is to utilize extrinsic, application specific metrics, such as student learning gains in an intelligent tutoring task; though a large sample size is needed to account for any potential variability related to such a measure.

7. Future Work

We are very encouraged that there is no statistical difference in evaluation of our system and the transcript-derived human tutoring turns, which we believe has the most comparable generation condition. In fact, the difference between our system's performance and that of well-trained tutors in these circumstances is less than 1/5th of a standard deviation.

These results indicate that there is still significant opportunity to refine our system's QG capabilities. Improving the system's ability to choose appropriate context should allow future evaluations to use the full tutorial contexts instead of only the student turns. Additionally, having gold standard semantic parses would allow us to better evaluate the quality of our dialog system.

Since the participants in this study are familiar with the questions produced by the system, we would also like to conduct a similar experiment using tutors who are not acquainted with our system to provide more independent judgments. Additionally, it is unclear what role the QtA pedagogy played in this evaluation. Conducting this experiment using a different tutorial style may help to clarify this question.

One of the most significant contributions of this paper is the light it has shed on issues involved in running a QG challenge.

Acknowledgements

We thank the tutors at Boulder Language Technologies for their help in this study. The research reported here was supported by The Institute of Education Sciences, U.S Department of Education grant R305B070008 and by the National Science Foundation grant R305B070434. The opinions expressed are those of the authors and do not represent views of IES, NSF, or the U.S. Department of Education.

References

[1] I. L. Beck, M. G. McKeown, J. Worthy, C. A. Sandora, and L. Kucan. Questioning the author: A year long classroom implementation to engage students with text. *The Elementary School Journal*, 96(4):387–416, 1996.

[2] Lee Becker and Wayne H. Ward. Adapting a frame-based dialogue manager for use in tutorial dialogues. Technical report, University of Colorado Boulder, 2009 (forthcoming).

[3] S. Issar and W. Ward. Cmu's robust spoken language understanding system. In *Eurospeech '93*, pages 2147–2150, 1993.

[4] Rodney D. Nielsen. Question generation: Proposed challenge tasks and their evaluation. In Vasile Rus and Art Graesser, editors, *Proceedings of the Workshop on the Question Generation Shared Task and Evaluation Challenge*, September 25-26 2008.

[5] Lawrence Hall of Science. Full option science system (foss). Nashua, NH, 2005.

[6] Natalie K. Person and Arthur C. Graesser. Human or computer? autotutor in a bystander turing test. In *ITS '02: Proceedings of the 6th International Conference on Intelligent Tutoring Systems*, pages 821–830, London, UK, 2002. Springer-Verlag.

[7] Lucy Vanderwende. The importance of being important. In Vasile Rus and Art Graesser, editors, *Proceedings of the Workshop on the Question Generation Shared Task and Evaluation Challenge*, September 25-26 2008.

An Empirically-Derived Question Taxonomy for Task-Oriented Tutorial Dialogue

Kristy Elizabeth BOYER[a], William J. LAHTI[a], Robert PHILLIPS[ab], Michael D. WALLIS[ab], Mladen A. VOUK[a], and James C. LESTER[a]
[a] *Department of Computer Science, North Carolina State University*
[b] *Applied Research Associates, Inc.*
Raleigh, North Carolina, USA
{keboyer, wjlahti, rphilli, mdwallis, vouk, lester}@ncsu.edu

Abstract. Devising an expressive question taxonomy is a central problem in question generation. Through examination of a corpus of human-human task-oriented tutoring, we have found that existing question taxonomies do not capture all of the tutorial questions present in this form of tutoring. We propose a hierarchical question classification scheme for tutorial questions in which the top level corresponds to the tutor's goal and the second level corresponds to the question type. The application of this hierarchical classification scheme to a corpus of keyboard-to-keyboard tutoring of introductory computer science yielded high inter-rater reliability, suggesting that such a scheme is appropriate for classifying tutor questions in design-oriented tutoring. We discuss numerous open issues that are highlighted by the current analysis.

Keywords. Question classification, Question taxonomies, Task-oriented tutoring, Tutorial dialogue

1. Introduction

The automatic generation of questions is an important emerging research area with potential utility for widespread applications [1]. One such application is natural language tutoring, in which questions are generated by an intelligent agent whose primary goal is to facilitate the learner's acquisition and construction of knowledge (*e.g.*, [2-9]). A tutor's pedagogical objectives may be accomplished with dialogue policies designed to enhance the learner's motivation, maintain an emotional state conducive to learning, or help the learner complete specific tasks relevant to the targeted knowledge or skill set.

A dialogue policy for question generation informs decisions about multiple features of an intelligent agent's conversational interactions. It governs decisions about the conditions under which a question should be posed, calibration of the level of content to be included, and choice of the tone of the realized question. Because a central feature involves determining the question type to be selected, devising an expressive question taxonomy is an important step toward high quality, robust automatic question generation [10, 11].

It is unlikely that a single question taxonomy can meet the needs of question generation for all application areas. In fact, even if we restrict our discussion to question generation for natural language tutorial dialogue, a single taxonomy is unlikely to suffice because, aside from the differences encountered across domains (*e.g.*, qualitative physics, English, mathematics), the format in which tutoring is conducted is likely to result in the need for different types of questions. For example, the tutoring sessions analyzed in this work demonstrate *task-oriented* tutoring, where the primary activity in which the learner engages is problem solving. In task-oriented tutoring, the tutor must be concerned with the quality of knowledge the student attains as expressed through the task at hand, and if a learning artifact is being designed, the tutor may also engage in question-asking specifically to address the quality of the artifact itself.

Question classification research has benefited several other fields of study, including computational modeling of question answering as a cognitive process [12] and answering students' questions with an intelligent tutoring system (*e.g.*, [13]). Recently, question taxonomies have been proposed that begin to address the needs of the question generation community [10, 11]. In this paper, we examine a corpus of human-human task-oriented tutoring and find that existing question taxonomies do not capture all the types of questions posed by the tutors. We propose an empirically-derived hierarchical question classification scheme in which the top level identifies the tutorial goal (*e.g.,* establish a problem-solving plan, scaffold the problem-solving effort through hinting). The second level of the hierarchy consists of annotation for question type; this level shares many categories with classification schemes proposed by Graesser *et al.* [10] and Nielsen *et al*. [11].

2. JavaTutor-Q Corpus

The JavaTutor-Q corpus of questions was collected across two semesters during tutoring studies. Participants were enrolled in a university introductory computer science class titled "Introduction to Computing – Java." The tutors and students interacted via remote keyboard-to-keyboard dialogue, with tutors viewing a real-time display of student problem-solving actions. Seventeen tutors were involved across the two studies; their experience level varied from one year of peer tutoring to several years of full classroom instruction. The majority of tutors, however, did not have any formal training or professional experience in tutoring or teaching; therefore, compared to studies of expert tutors (*e.g.*, [4, 14]), the tutors under consideration here are unskilled. Eighty-two participants interacted for one session each, each session lasting approximately one hour. The complete corpus contains a total of 10,179 utterances. Tutors contributed 6,558 of these utterances, of which 714 were identified as questions during previous dialogue act tagging efforts [15, 16].[1] This corpus of questions serves as the basis for the question classification scheme presented here.

The JavaTutor-Q corpus arose from naturalistic keyboard-to-keyboard human tutoring of introductory computer science: that is, tutors were given no specific

[1] Initially there were 721 questions; however, during the tagging process reported here, 7 of these were identified as non-questions whose original dialogue act tag was erroneous.

instructions regarding tutoring strategies.[2] Qualitative exploration of the corpus revealed an important phenomenon that has shaped the question taxonomy presented here. Table 1 illustrates that tutors in this study often provide hints in the form of questions. This behavior is likely an example of a polite strategy that allows the student to "save face" in the presence of a mistake [17, 18]. Although an indirect approach may not always be ideal for student learning [19], a taxonomy of tutorial questions should capture indirect approaches. The subsequent choice of whether to implement these tactics can then be treated as a higher-level design decision.

Table 1. Excerpts from the JavaTutor-Q Corpus

Tutor 1:	So... parseInt takes a String and makes it into an int... but we only want one digit, so how are we going to get just one digit as a string? *[Proc]*
Student 1:	charAt?
Tutor 1:	Well that would give us a char.
Tutor 1:	There's another String operation that can give us part of a string as a string... I think it's subString? *[Hint]*
Student 1:	Right.
Student 2:	[Declares five distinct variables in the problem-solving window]
Tutor 2:	We can approach this using five distinct variables, but when we work with them in our loops to draw the bar codes, I'm wondering whether making an array will be a better alternative? *[Hint]*
Student 2:	Yeah, we could. Would make looping easier too.

3. Hierarchical Question Annotation

At its top level, the proposed question taxonomy intends to capture the tutorial goal that motivated each question. At its second level, this taxonomy captures the question type, a distinction that is more closely related to the surface form of the question.

3.1. Level 1: Tutorial Goal

It has been recognized that tagging a human-human corpus with tutorial goals can inform the design of the natural language generation component of tutoring systems. For example, the NLG component of CIRCSIM-Tutor was based partly on the

[2] Tutors were provided with a suggested dialogue protocol for maintaining anonymity in the event that the student directly inquired about the tutor's identity.

annotation of tutorial goals [20]. The detailed hierarchical annotations used for CIRCSIM-Tutor were not directly reusable for our purposes because many of the tutoring techniques present in their corpora of expert tutoring were not present in the JavaTutor-Q corpus. In addition, our current goal is to focus specifically on tutorial questions. To that end, we propose a new set of tutorial goals that is intended to capture what goal motivated the tutor to ask each question.

Corbett & Mostow [21] suggest that ideal questions in a reading comprehension tutor should address tutorial goals such as 1) assessing comprehension of text, 2) assessing student engagement, 3) evaluating tutor interventions, 4) providing immediate feedback, 5) scaffolding comprehension of text, 6) improving student engagement, and 7) scaffolding student learning. Some of these goals have direct analogy for task-oriented tutoring; for example, "scaffolding comprehension of text" becomes "scaffolding the student's problem-solving effort." Table 2 presents our set of tutorial goals, which began with analogues to the above goals and then evolved iteratively through collaborative tagging by two annotators. Specifically, an initial set of goals was informed by qualitative inspection of the corpus, and then goals were added or merged until both annotators felt that all tutorial questions in the "training" sample of approximately 400 questions were well-represented. After finalizing the tutorial goal tags, the first annotator tagged all of the remaining questions, for a total of 714 utterances. A second annotator tagged a subset of tutoring sessions, totaling 118 questions, that were not part of the training set. The resulting unweighted Kappa agreement statistic was 0.85, indicating high reliability of the tutor goal annotation scheme [22].

3.2. Level 2: Question Type

The second level of annotation was performed after tutor goal tagging had been completed and disagreements between annotators had been resolved collaboratively. In the second phase, each question was classified according to its type. The question type classification scheme relies heavily on the question taxonomy proposed by Nielsen *et al.* [11], which itself was informed by Graesser *et al*. [10]. The process of formulating the current set of question types was analogous to the formulation of the tutorial goal set. We began with the union of question types from [11] and [10], and through collaborative tagging of a training set of approximately 450 questions, this set was refined until both annotators felt all training questions were adequately classified. Table 2 illustrates the resulting question classification scheme. The first annotator tagged the entire question corpus, while a second annotator applied the question classification scheme to 117 questions that were not part of training. The resulting unweighted Kappa statistic of 0.84 indicates high reliability of the classification scheme.

4. Discussion and Open Issues

Understanding question types is an important step toward the robust automatic generation of high-quality questions. This paper has presented a two-level classification scheme for tutorial questions that occurred as unskilled human tutors worked with novice computer science students who were designing and implementing the solution to an introductory computer programming problem. In the study presented

Table 2. Tutorial Goals (Level 1) and Co-occurring Question Sub-Types (Level 2)

Tutorial Goal	Freq (n_{total} = 714)	Details	Question Sub-Types[3]
Plan	164	Establish a problem-solving plan. Ascertain what the student wants, prefers, or intends to do.	Definition, Free Creation, Feature or Concept Completion, Free Option, Goal, Judgment, Justification, Planning, Procedural, Status
Ascertain Student's Knowledge	282	Find out whether the student knows a specific factual or procedural concept.	Causal Antecedent, Calculate, Causal Consequence, Definition, Enablement, Feature/Concept Completion, Free Option, Improvement, Justification, Knowledge, Procedural, Quantification, Status
Hint	127	Scaffold the student's problem-solving effort.	Hint, Causal Consequence
Repair Communication	34	Disambiguate or correct a previous utterance in the dialogue.	Clarification, Feature/Concept Completion
Confirm Understanding	73	Confirm the student's understanding of a previous utterance in the dialogue or of a previously-scaffolded problem-solving step.	Assess, Backchannel, Causal Antecedent, Confirmation, Status
Engage Student	14	Elicit an utterance from the student, either at the beginning of the tutoring session or after a prolonged period of student silence.	Feature/Concept Completion, Goal, Status
Remind/Focus	20	Focus the student's attention on a previous utterance or problem-solving step for instructional purposes.	Assess, Feature/Concept Completion, Focus, Hint, Procedural

here, the tutors' goals were annotated by researchers in a *post hoc* manner. The informativeness of this tagging might be enhanced by future work in which the tutors themselves indicate the goal of each question either *post hoc* or, perhaps preferably, in real time. Ascertaining the tutors' local goal for each question, along with the state information that motivated that goal, would provide valuable insight for future automatic tutorial question generation systems.

As illustrated in Table 3, several question types from existing taxonomies did not occur in the current corpus. This phenomenon is likely due to the skill level of the tutors; they often utilized very broad question types, such as *Confirmation*, which rely heavily on the student's ability to self-assess [19]. The difference in types of questions asked by experts and novices is an important distinction (*e.g.*, [14, 23]), but because no conclusive differences in effectiveness have been established among question types, it

[3] These sets of question sub-types were not formulated *a priori*; rather, this column displays all question types that occurred in combination with each tutor goal after the question annotation was complete.

Table 3. Question Types (Level 2)

Question Type	Examples	Freq. (n_{total} = 714)	Source		
			Graesser *et al.*	Nielsen *et al.*	New
Assessment	Do you think we're done?	6			●
Backchannel	Right?	6			●
Calculation	What is 13 % 10?	11		●	
Causal Anteced.	Why are we getting that error?	2	●	●	
Causal Conseq.	What if the digit is 10?	8	●	●	
Clarification	What do you mean?	31			●
Composition	*Not present in the current corpus.*	0		●	
Comparison	*Not present in the current corpus.*	0	●	●	
Confirmation	Does that make sense?	60			●
Feature/Concept Completion	What do we want to put in digits[0]?	109	●	●	
Definition	What does that mean?	2	●	●	
Disjunctive	*Subsumed by other tags in current corpus.*	0	●		
Enablement	How are the digits represented as bar codes?	2	●	●	
Example	*Not present in the current corpus.*	0	●	●	
Expectation	*Not present in the current corpus.*	0	●		
Focus	See where the array is declared?	11			●
Free Creation	What shall we call it?	1		●	
Free Option	Should the array be in this method or should it be declared up with the other private variables?	6		●	
Goal Orientation	Did you intend to declare a variable there?	20	●	●	
Hint	We didn't declare it; should we do it now?	128			●
Improvement	Can you see what we could do to fix that?	9		●	
Interpretation	*Not present in the current corpus.*	0	●	●	
Judgment	Would you prefer to use math or strings?	17	●	●	
Justification	Why are we getting that error?	3		●	
Knowledge	Have you ever learned about arrays?	93			●
Procedural	How do we get the i^{th} element?	127	●	●	
Quantification	How many times will this loop repeat?	3	●	●	
Status	Do you have any questions?	17			●
Verification	*Subsumed by other tags in current corpus.*	0	●		

is important for question taxonomies, especially at the formative stages of research in automatic question generation, to be comprehensive. Proceeding from that foundation, investigating the effectiveness of question types given such features as the tutor's immediate goal and knowledge of the student and the problem-solving state will be an important direction for future work. Finally, it is important to note that in the question

classification project presented here, questions were tagged in their original dialogue context. The annotators felt it was often important to consider the surrounding context (usually the previous two or three utterances) for both tutorial goal annotation and question type tagging. A rigorous study of the importance of context for question classification could shed light on how much context is necessary for a question generation system to make sound decisions.

5. Conclusion

Question classification schemes have been the topic of research for several decades, and increased interest in the task of automated question generation raises new issues that highlight the importance of empirically-grounded question taxonomies. We have proposed a hierarchical question classification scheme designed to capture the phenomena that occur during human-human task-oriented tutoring. In the first level of the proposed taxonomy, questions are classified according to the tutor's goal, an approach inspired by previous work using tutorial goal annotation to inform the natural language generation of tutoring systems. The second level of the scheme captures the realized question type using an augmented version of existing question taxonomies. Both levels of question classification were applied with very high inter-rater reliability. This classification scheme represents a first step toward a comprehensive question taxonomy for task-oriented tutoring.

Acknowledgments

This research was supported by the National Science Foundation under Grants REC-0632450, IIS-0812291, CNS-0540523, and GRFP. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] V. Rus and A. Graesser. *The Question Generation Shared Task and Evaluation Challenge.* In press.

[2] A. Graesser, G. Jackson, E. Mathews, et al. Why/AutoTutor: A Test of Learning Gains from a Physics Tutor with Natural Language Dialog, *Proceedings of the Twenty-Fifth Annual Conference of the Cognitive Science Society,* pp. 1-6, 2003.

[3] C. Zinn, J. D. Moore and M. G. Core. A 3-tier Planning Architecture for Managing Tutorial Dialogue, *Proceedings of the 6th International Conference on Intelligent Tutoring Systems,* pp. 574-584, 2002.

[4] M. Evens and J. Michael. *One-on-One Tutoring by Humans and Computers.* Mahwah, New Jersey: Lawrence Erlbaum Associates, 2006.

[5] V. Aleven, K. Koedinger and O. Popescu. A Tutorial Dialog System to Support Self-explanation: Evaluation and Open Questions, *Proceedings of the 11th International Conference on Artificial Intelligence in Education,* pp. 39-46, 2003.

[6] D.J. Litman, C.P. Rosé, K. Forbes-Riley, K. VanLehn, D. Bhembe and S. Silliman. Spoken Versus Typed Human and Computer Dialogue Tutoring, *International Journal of Artificial Intelligence in Education,* vol. 16, iss. 2, pp. 145-170, 2006.

[7] H.C. Lane and K. VanLehn. Teaching the Tacit Knowledge of Programming to Novices with Natural Language Tutoring, *Computer Science Education,* vol. 15, iss. 3, pp. 183-201, 2005.

[8] E. Arnott, P. Hastings and D. Allbritton. Research Methods Tutor: Evaluation of a Dialogue-Based Tutoring System in the Classroom, *Behavior Research Methods,* vol. 40, iss. 3, pp. 694-698, 2008.

[9] K. VanLehn, P.W. Jordan, C.P. Rose, et al. The Architecture of Why2-Atlas: A Coach for Qualitative Physics Essay Writing, *Proceedings of the 6th International Conference on Intelligent Tutoring Systems,* vol. 2363, pp. 158–167, 2002.

[10] A. Graesser, V. Rus and Z. Cai. Question Classification Schemes, *Proceedings of the 1st Workshop on Question Generation,* 2008.

[11] R. Nielsen, J. Buckingham, G. Knoll, B. Marsh and L. Palen. A Taxonomy of Questions for Question Generation, *Proceedings of the 1st Workshop on Question Generation,* 2008.

[12] W. Lehnert. *The Process of Question Answering - A Computer Simulation of Cognition.* Hillsdale, New Jersey: Lawrence Erlbaum Associates, 1978.

[13] L. Acker, J. Lester, A. Souther and B. Porter. Generating Coherent Explanations to Answer Students' Questions. In H. Burns, J.W. Parlett, et al. (Eds.), *Intelligent Tutoring Systems: Evolutions in Design,* 151-176. Hillsdale, New Jersey: Lawrence Erlbaum Associates, 1991.

[14] W. Cade, J. Copeland, N. Person and S. D'Mello. Dialog Modes in Expert Tutoring, *Proceedings of the 9th International Conference on Intelligent Tutoring Systems,* pp. 470-479, 2008.

[15] K.E. Boyer, M. A. Vouk and J. C. Lester. The Influence of Learner Characteristics on Task-Oriented Tutorial Dialogue, *Proceedings of the 13th International Conference on Artificial Intelligence in Education,* pp. 365-372, 2007.

[16] K.E. Boyer, R. Phillips, M. Wallis, M. Vouk and J. Lester. Balancing Cognitive and Motivational Scaffolding in Tutorial Dialogue, *Proceedings of the Proceedings of the 9th International Conference on Intelligent Tutoring Systems,* pp. 239-249, 2008.

[17] P. Brown and S. Levinson. *Politeness: Some Universals in Language Usage.* Cambridge University Press, 1987.

[18] N. Wang, W. L. Johnson, P. Rizzo, E. Shaw and R. E. Mayer. Experimental Evaluation of Polite Interaction Tactics for Pedagogical Agents, *Proceedings of the 10th International Conference on Intelligent User Interfaces,* pp. 12-19, 2005.

[19] N.K. Person, R.J. Kreuz, R.A. Zwaan and A.C. Graesser. Pragmatics and Pedagogy: Conversational Rules and Politeness Strategies may Inhibit Effective Tutoring, *Cognition and Instruction,* vol. 13, iss. 2, pp. 161-188, 1995.

[20] J.H. Kim, R. Freedman, M. Glass and M.W. Evens. Annotation of Tutorial Dialogue Goals for Natural Language Generation, *Discourse Processes,* vol. 42, iss. 1, pp. 37-74, 2006.

[21] A. Corbett and J. Mostow. Automating Comprehension Questions: Lessons from a Reading Tutor, *Proceedings of the 1st Workshop on Question Generation,* 2008.

[22] J.R. Landis and G. Koch. The Measurement of Observer Agreement for Categorical Data, *Biometrics,* vol. 33, iss. 1, pp. 159-174, 1977.

[23] M. Glass, J. H. Kim, M. W. Evens, J. A. Michael and A. A. Rovick. Novice vs. Expert Tutors: A Comparison of Style, *Proceedings of the 10th Midwest Artificial Intelligence and Cognitive Science Conference,* pp. 43-49, 1999.

Generating Questions Automatically from Informational Text

Wei CHEN[1], Gregory AIST, and Jack MOSTOW
Project LISTEN, School of Computer Science, Carnegie Mellon University

Abstract. Good readers ask themselves questions during reading. Our goal is to scaffold this self-questioning strategy automatically to help children in grades 1-3 understand informational text. In previous work, we showed that instruction for self-questioning can be generated for narrative text. This paper tests the generality of that approach by applying it to informational text. We describe the modifications required, and evaluate the approach on informational texts from Project LISTEN's Reading Tutor.

Keywords. Question generation, informational text, self-questioning, reading tutor, comprehension strategy instruction

Introduction

Good readers ask themselves questions during reading. Based on comprehension gains, self-questioning was the most effective reading comprehension strategy identified by the National Reading Panel [1]. So it would be useful for an intelligent tutor to automatically generate instruction for the self-questioning strategy to help students understand text. Ultimately we would like to generate effective self-questioning instruction automatically from any given text, focusing on children's text.

Previous work [2] used a two-step approach for generating instruction to model and scaffold the self-questioning strategy: first generate questions from the text, and then augment the questions into strategy instruction. It showed how to generate questions automatically from narrative text. Here we test the generality of that approach by extending it to another important genre: informational text.

Informational text is an important source of knowledge. Reading researchers have found that even young children can benefit from it, if taught the right strategy [3, 4].

Compared to narrative fiction, informational texts have different text structure and serve different reading goals [5]. For example, sentences (1) and (2) came from narrative and informational text, respectively.

(1) Peter thought it best to go away without speaking to the white cat.
(2) Rainbows are seen after it rains and the sun is out.

[1] Corresponding Author. The research reported here was supported by the Institute of Education Sciences, U.S. Department of Education, through Grant R305B070458. The opinions expressed are those of the authors and do not necessarily represent the views of the Institute and the U.S. Department of Education. We also thank Nell Duke for her expertise, comments, and advice.

As exemplified by sentence (1), narrative text involves characters, their behavior, and mental states that drive it. In contrast, informational text does not require characters. In addition, it places more emphasis on descriptions and explanations, which are often used to introduce objective phenomena, as in sentence (2).

The example below consists of a paragraph from the informational text "Life under the Sea Part 1 – the Meaning of Life," followed by a question generated from the text.

Text: What does it mean to be "alive?" What is the difference, say, between an elephant and a boulder? This seems to be an obvious question but one that may be difficult to answer. All living things are not exactly alike. For example, not all living things breathe air, or have blood, or grow hair, like we do. Likewise, we can't live under water like fish do.

Question: Why can't we live under water like fish do?

The rest of this paper is organized as follows. Section 1 summarizes our previous work on question generation instruction from narrative text. Section 2 describes how we extend that work to handle informational text. Section 3 presents evaluation criteria and results. Section 4 discusses the generality and importance of the approach based on the evaluation results. Section 5 summarizes the contribution, lists some limitations of the approach, and points out future work.

1. Question generation for narratives

Our question generation task sits in the context of generating instruction for the reading comprehension strategy of self-questioning. The instruction includes four phases: describe, model, scaffold, and prompt the self-questioning strategy. Of these phases, modeling and scaffolding the strategy rely on question generation.

Previous work [2] described how to generate questions from narrative text and convert questions into modeling and scaffolding instruction. Given a piece of text, our question generation system transforms it into a situation model. The model of mental states is a partial simulation of the student's own "theory of mind," and thus the method creates a situation model of the textbase. A mental state expression (e.g., "think," "regret") indicates an opportunity to prompt strategy instruction. To model the self-questioning strategy for the student, the system transforms the sentence into a question, e.g., "Why did Peter think it best to go away without speaking to the white cat?" To scaffold the strategy, the system leads the student to construct a question by choosing a character, a question type, and a completer. For example,

Tutor: *Let's make a question about _____ .*
(Peter; Mr. McGregor; the old mouse; the white cat)
Student: [The student chooses *Mr. McGregor* from an on-screen menu of 4 characters.]
Tutor: *Let's ask a ___ (what; why; how) question.*
Student: [The student chooses *why*.]
Tutor: *Great! 'Why' questions can help you understand by making you think!*
Tutor: *Let's complete your question: Why did Mr. McGregor ___ (try to put his foot upon Peter; try to find his way straight across the garden; think it best to go away without speaking to the white cat)?*

Student: [The student chooses *try to find his way straight across the garden*.]
Tutor: *I'm not sure Mr. McGregor tried to find his way straight across the garden. Would you like to try again?*

The character and completer choices are all extracted from the story. Constructible questions include system-generated questions and other combinations such as "How did Mr. McGregor try to put his foot upon Peter?" and "What did the white cat think?." "What" questions trigger different completers, not shown in the example.

How general is our question generation mechanism? We test it here by adapting it to informational text. We could not use exactly the same approach for informational text, because of its different text structure and vocabulary [6]. Therefore, we had to add knowledge to our question generation system to deal with two specific issues in informational text: the "where" of decisions about what sentences to use to generate questions; and the "how" of rules used for building the situation model, and question templates to map the text into questions.

2. Locating question opportunities in informational text

We generated questions of the same forms as for narrative text, and also of new forms.

2.1. Mental states in informational text

Our previous work relied on mental states to locate possible questioning points. By "mental states," we mean beliefs, states of knowledge, points of view, or suppositions. However, mental states are not as central in informational texts as in narrative texts, in terms of their role in understanding the main idea of the text. Using the same set of mental state expressions, we found that mental states occurred 1382 times in 162 narrative texts (8.5 mental expressions per text) from Project LISTEN's Reading Tutor [7] and 727 times in the 105 informational texts that we used as our training data (6.9 modal terms per text). This difference suggests that words and phrases indicating mental states occur more frequently in narrative text than in informational text, although the gap does not seem very big.

On the other hand, mental states may vary by text genre in terms of what relationships between clauses they represent (e.g., causal vs. coincidental vs. mood-setting). In narrative text, mental states are normally attached to a character in the story, as in "Peter thought." Moreover, mental states of a character tend to reveal the motivation of the character and thus are likely to indicate causal relationships to events involving the character. In contrast, informational text may not contain any character in the same sense as in narrative stories. However, it may still refer to agents' mental states (**boldfaced** here) as the motive force or result of some events or phenomena, e.g.:

(3) Fish have "noses" (called nares) that don't look anything like our own, yet their **purpose** is to smell chemicals in the water.

Informational text may also refer to mental states of people outside the text, such as the reader or author, e.g.:

(4) If you're an American citizen 18 years of age or older, you probably **think** you have the right to vote for presidential candidates in the national election.

Similarly, it may refer to beliefs of authoritative sources or the general public, e.g.:

(5) It is **thought** that they use this structure to detect prey, perhaps being able to distinguish the weak electrical signals given off by injured animals.

Thus mental state expressions appear in some – but not all – informational text.

2.2. Extension to other categories of question indicators in informational text

Based on our observations, we believe that using mental states as the only indicators of good questions will not suffice for informational text. Our criteria for selecting questioning indicators are that the indicator has to signal key information about the text and it should be feasible for an intelligent tutor to recognize and generate good questions. According to expert pedagogy, teaching text structure is important in comprehending informational text [e.g. 8, 9]. However, figuring out global text structure automatically is still an active research problem that has not been solved completely, so we started with discourse markers that indicate causal relationships (i.e., conditions and temporal context) and modality (i.e., possibility and necessity).

2.2.1. Causality: conditional and temporal contexts

Conditional and temporal context are very important in informational text. Compared to narratives, conditional context and temporal context in informational text are more likely to describe causation. For example, in sentence (2), the temporal expression "after it rains and the sun is out" describes a causal condition of the formation of a rainbow. Here is another example of conditional context (shown in **boldface**):

(6) **If humans removed all the kelp from the sea** soon all the other sea life would start to suffer as well.

To search for linguistic expressions that indicate conditional contexts, we enumerated 4 words and constructions we noticed in the training data as questioning points, namely "if," "even if," "only if," and "as long as," which occurred 37 times in the training data.

To find temporal expressions, we used the ASSERT semantic role labeler [10] to annotate the corpus. Then our system looks for expressions marked by the ARGM-TMP tag [11] for "temporal expression." The system found 763 such temporal expressions in the training data. We noticed four kinds of temporal expressions in our training data: a general condition such as "after it rains and the sun is out," a date or time such as "in 1999," a duration of time such as "for several hours," and a rhetorical relationship (at the same time) such as "while she was reading." Here we focus only on the first type of temporal expression, which tends to indicate causality. To filter out the other three types of temporal expressions, we used regular expressions to detect dates, duration (i.e., started with the word "for"), and expressions that indicate things happening at the same time (i.e., started with the word "while"). We also noticed that some words about frequency such as "usually" and "sometimes" can lead to trivial "when"-questions, and they are often tagged individually with ARGM-TMP as in "[ARGM-TMP usually]," which is not as informative for our purpose of finding causality. To filter them out, we used a heuristic, namely ignore temporal expressions that contain only one word. This heuristic filtered out 35.8% (273) of the temporal expressions, yielding 490 questioning points about temporal contexts.

2.2.2. Linguistic modality: possibility and necessity

Linguistic modality such as possibility and necessity is also important in informational text. Linguistic modality is often expressed by auxiliary verbs. The most frequent auxiliary verbs can be hypothetical (e.g. "would"), predictive (e.g. "will"), or prescriptive (e.g. "should," "ought to," "must"). In sentence (7) below, the word "should" expresses goats' need for covered shelters. Thus a reasonable question to generate from this sentence is "Why should goats have covered shelters?"

(7) All goats **should** have covered shelters where they can escape the weather.

We identified 8 auxiliary verbs and constructions from the training data to extract modality patterns, including "would," "will," "should," "shall," "could," "ought to," "must" and "may." These constructions appeared 179 times in our training data.

2.3. Question generation process for informational text

Our system generates questions from the situation model, which it constructs using schema-building rules. The question generation system uses one rule for each type of target conditional, temporal or modality expression. Based on semantic categories of the target expressions, we defined 6 rules, which build various sub-contexts and store elements of statements in a situation model. For example, one schema-building rule for modeling temporal context can be paraphrased as "create a temporal context to store the when-statement; re-order existing temporal contexts based on time order."

We added 4 question templates to transform the information retrieved from situation models into questions. The question template for conditional context is "What would happen if <x>?" For temporal context, we used two templates: "When would <x>?" and "What happens <temporal-expression>?" For linguistic modality, we used "Why <auxiliary-verb> <x>?" Here <x> maps to semantic roles tagged with ARG0 (the agent), TARGET (the verb), ARG1 (the theme), and ARG2, if any. Since we aimed at questions about general conditions, which do not concern tense, we included auxiliary verbs such as "would" in the question templates. Therefore, we do not need morphology generation for verbs, as we did for narrative text questions. Table 1 shows questions generated from sentences (2), (6) and (7).

Table 1. Questions generated from temporal, conditional, and modality expressions.

Sentence number	Resulting question
(2)	a. When would rainbows be seen? b. What happens after it rains and the sun is out?
(6)	What would happen if humans removed all the kelp from the sea?
(7)	Why should all goats have covered shelters?

3. Results

We evaluated the quality of the generated questions by the same criteria we used for mental state questions, i.e., the question had to be grammatically correct and it had to make sense in the context of the text. These criteria describe plausible candidates that we considered worth showing to experts for review. To evaluate our approach, we used a separate set of 26 informational texts from the Reading Tutor as our test data, which did not overlap with the training data. The test data contained 444 sentences.

Table 2 summarizes the evaluation statistics and results. We hand-evaluated the questions in each of the three categories. To validate the evaluation result, we would have another rater and calculate inter-rater agreement.

Questions about conditional context can be classified into two kinds, depending on the semantic role of if-clauses. In the test data, three if-clauses turned out to be direct objects, as in "Scientists wondered **if meat-eating Tyrannosaurus rex had ever eaten Triceratops**." Others were adverbs, as in "**If humans removed all the kelp from the sea** soon all the other sea life would start to suffer as well." The implausible conditional questions were caused by unresolved coreference and ambiguity of "if" under different contexts. For example, the sentence "**If so**, then you have eaten kelp" resulted in an implausible question "What would happen if so?" by failing to resolve what "so" refers to. Also, some phrases like "as if" changed the meaning of "if" which in our case was defined to set a conditional context. The sentence "Sit beside a quiet pool of water and you'll soon see water striders skating as **if on ice**" resulted in the out-of-context question "What would happen if on ice?"

Questions about temporal information were rated lowest in terms of plausibility. 66.7% (20) of the implausible questions were due to parsing errors. For example, in the parsing result "If the pressure changes over a large area it can cause [ARG1 winds] to [TARGET blow] [ARGM-TMP in a huge circle]," the tagger erroneously tagged "in a huge circle" as a temporal expression, leading to the implausible question "What would happen when in a huge circle?" 33.3% (10) of the implausible questions came from undetected constructions that do not belong to the first type of temporal expressions, such as "at present" and "some day." For example, from the sentence "**At present** totem poles are sold to people who collect them and to museums," a question was "When would totem poles be sold to people who collect them and to museums?," which is not asking something that the sentence is intended to convey.

All the implausible modality questions we observed were caused by parsing errors (including coreference and negation errors). We use semantic roles as parameters to build the situation model, but sometimes the semantic roles are only partially tagged. For example, in "[ARG0 Skin cells] [ARGM-MOD must] [ARGM-DIS also] [TARGET make] [ARG1 sure] to keep harmful things out of the body," the incomplete semantic role labeling led to the partial question "Why must skin cells make sure?"

Table 2. Evaluation Results

Question type	Number of matched linguistic patterns	Number of generated questions	Percentage of plausible questions
Condition	15	15	86.7% (13/15)
Temporal information	44	88	65.9% (58/88)
Modality	33	77	87.0% (67/77)

4. Discussion

The goal of this paper is to extend our question generation approach for narrative fiction to handle informational text. This problem involves two issues: a) how well does the approach work on informational text? b) how much additional work does it take to extend question generation from narrative to informational text? Section 3 reported the quality of questions generated by the system. During the evaluation, we have noticed that some generated questions may not have explicit answers in the text,

such as if-clauses as the direct object of a verb (e.g. "What would happen **if meat-eating Tyrannosaurus rex had ever eaten Triceratops**?"). This property makes the question itself interesting insofar as it gets the student to think about a possible result that could be caused by the condition, and the answers may not be obvious from the text. Similar to the case in narratives, the schema-building rules we used for informational text can be used for extracting answers and detecting questions with false premises, which is helpful for providing feedback to students in a complete instruction scenario. To adapt our approach to informational text, we kept the question generation process and same language technology tools, and we added three types of knowledge. Table 3 compares the knowledge we used for the two genres.

Generating good questions requires inference, which is a natural language understanding problem. We know that natural language understanding is "AI-complete" because of the inference problem. We do not attempt to solve the entire inference problem, but to identify some inferences that we know how to make. At the knowledge representation level, we built only partial situation models (i.e., about conditional and temporal context and modality). We looked for types of inferences that are feasible to extract and do not rely on world knowledge beyond the sentence (or story). The only information we needed for capturing important question indicators was knowledge of discourse markers such as if-constructions, temporal expressions, and auxiliary verbs.

Table 3. Comparison of question generation for informational text and narrative text.

Genre	Linguistic patterns	Type of questions	Generation templates
Narrative	mental state expressions	"What," "Why" and "How" questions about mental states	What did <character> <verb>?
			Why/How did <character> <verb> <complement>?
			Why was/were <character> <past-participle>?
Informational text	if-constructions	"What-would-happen-if" question about conditional context	What would happen if <x>?
	temporal expressions	"When-would-x-happen" question about temporal context	When would <x> happen?
		"What- happens-when" question about temporal context	What happens <temporal-expression>?
	auxiliary verbs	"Why" question about possibility and necessity	Why <auxiliary-verb> <x>?

5. Conclusion, Limitations and Future Work

In this paper, we tested the generality of our question generation approach by extending it to another genre: informational text. We described an approach to generate questions from informational text, which could then be used to generate modeling and scaffolding instruction for the reading comprehension strategy of self-questioning. We extended the question generation approach to informational text by adding three types of knowledge: a) discourse markers for locating opportunities for questions; b) schema-building rules for managing information in a situation model; c) question templates for converting information into questions. We proposed three types of questions for informational text: questions about conditional context, questions about temporal information, and questions about possibility and necessity. We also

demonstrated how discourse markers, such as conjunctions and certain kinds of verbs, can be used as indicators of places to ask questions about text.

So far, we covered only three types of questions to generate from informational text. There are many other important features of informational text that can cause difficulty for young children, such as its non-linear text structure and implicit causality. In this paper, we explored discourse markers for causal implication. Future work includes extending the existing approach to include inference rules that can automatically discover implicit logical relationships in the text and build global text structures in order to generate other important questions (and their answers).

We showed the automatically generated questions from one example story to a reading expert for evaluation. Although the expert did not raise grammatical issues about the questions, she felt that most of them lacked pedagogical value. This result was surprising to us because the questions generated for narrative fiction had fared far better. In future work, we will try to find out what caused similar approaches to yield different pedagogical value in narrative fiction and informational text. We will also look for more educationally beneficial types of questions to generate.

References (Project LISTEN publications are at www.cs.cmu.edu/~listen)

[1] NRP. Report of the National Reading Panel. Teaching children to read: An evidence-based assessment of the scientific research literature on reading and its implications for reading instruction. 2000, http://www.nichd.nih.gov/publications/nrppubskey.cfm: Washington, DC.

[2] Mostow, J. and W. Chen. Generating Instruction Automatically for the Reading Strategy of Self-Questioning. *The 14th International Conference on Artificial Intelligence in Education* 2009. Brighton, UK.

[3] Duke, N.K., V.S. Bennett-Armistead, and E. Roberts, eds. *Incorporating Informational Text in the Primary Grades*. Comprehensive Reading Instruction Across the Grade Levels, ed. C. Roller. 2002, DE: International Reading Association: Newark.

[4] Moss, B. Teaching Expository Text Structures through Information Trade Book Retellings: Teachers Can Help Students Understand Common Expository Text Structures by Having Them Retell Information Trade Books. *The Reading Teacher*, 2004. *57*.

[5] Meyer, B.J.F. Prose analysis: Purposes, procedures, and problems. In C.C.B.M. Pressley, Editor, *Understanding expository text*, 11-64. Erlbaum: Hillsdale, NJ, 1985.

[6] Purcell-Gates, V. and N.K. Duke. Explicit explanation/teaching of informational text genres: A model for research. *Crossing Borders: Connecting Science and Literacy conference* 2001. Baltimore, MD.

[7] Mostow, J. and J. Beck. When the Rubber Meets the Road: Lessons from the In-School Adventures of an Automated Reading Tutor that Listens. *Conceptualizing Scale-Up: Multidisciplinary Perspectives* 2003. Park Hyatt Hotel, Washington, D.C.

[8] Duke, N.K. and V.S. Bennett-Armistead. *Reading & Writing Informational Text in the Primary Grades: Research-Based Practices*. 2003: Teaching Resources.

[9] Anderson, E. and J.T. Guthrie. Motivating children to gain conceptual knowledge from text: The combination of science observation and interesting texts. *The Annual Meeting of the American Educational Research Association* 1999. Montreal, Canada.

[10] Pradhan, S.S., W. Ward, K. Hacioglu, J.H. Martin, and D. Jurafsky. Shallow Semantic Parsing using Support Vector Machines. *the Human Language Technology Conference/North American chapter of the Association for Computational Linguistics annual meeting (HLT/NAACL-2004)* 2004. Boston, MA.

[11] Palmer, M., D. Gildea, and P. Kingsbury. The Proposition Bank: A Corpus Annotated with Semantic Roles. *Computational Linguistics Journal*, 2005. *31*(1).

Question Generation: Taxonomies and Data

Corina FORĂSCU[a, b,1] and Iuliana DRĂGHICI[a]
[a] *Faculty of Computer Science, University "Al. I. Cuza" of Iaşi, Romania*
[b] *Research Institute in Artificial Intelligence, Romanian Academy, Bucharest*
{corinfor, idraghici}@info.uaic.ro

Abstract. Question Generation based on various text collections is an essential component in learning processes, dialogue systems (diagnosis and treatment in medicine or legal cross-examinations) or evaluation campaigns such as TREC or CLEF. With the objective of building a QG prototype, the paper presents three important question taxonomies, at the basis of other three question taxonomies used in various domains like critical thinking assessment and evaluation campaigns.

Keywords. question taxonomies, data for question generation

Introduction

A clear inventory of some of the most used question types, together with adequate textual data from where the questions and their answers to be found, are an important step in a short-term QG campaign; in Computational Linguistics, Artificial Intelligence, Education, and Discourse Processing, question taxonomies have been developed and exploited (Rus & Graesser, 2009).

The formalizations and models obtained from question taxonomies are useful in an automatic question-generation (QG) system, since they can constitute a model for question creation and extraction. The collection of questions from which the taxonomy was inferred, together with their answers, if available, can be also used for the evaluation of an automatic QG system. Starting with a collection of texts and according to a specific purpose, a selected taxonomy of questions is used to feed the system. The questions are generated from the text corpus (minimally tokenized, POS-tagged and chunked), using a lexicalized model of the taxonomy and a set of pattern-rules, previously developed either manually or automatically. Such a system that works (semi)automatically would benefit not only research-oriented tasks such as QA, but also it could be useful in the automatic creation of questionnaires, quizzes and test questions.

The paper gives in the first section a brief overview of important theoretical question taxonomies (Socrates, Bloom, Lehnert and Graesser). Their use in practical settings (LSAT tests, TREC and CLEF QA tracks) is presented in the second section of the paper. The paper ends with main conclusions and future work.

[1] Corresponding Author: corinfor@info.uaic.ro.

1. Theoretical approaches in question taxonomies

1.1. The taxonomy of Socratic questions

The taxonomy of Socratic questions (Paul, 1993) represents a special kind of hierarchy, without following a pattern or design. The Socratic approach is based on the practice of thoughtful questioning that Socrates believed enables students to examine ideas in a logical way and lead them to reliable knowledge construction; therefore the Socratic Questions are deep questions (Rus, Graesser, 2009). These six questions are essential for critical thinking[2]; in real life they are used in teaching, (medical) examination, engineering and trial questioning (Paul, 1993).

1. `Clarification` - gets the interlocutor to think deeper about the subject at hand and prove the concept behind his argument.
2. `Probing assumptions` - makes the interlocutor think about suppositions and unquestioned assumptions on which he is founding his argument.
3. `Probing Reasons and Evidence` - digs into the reasoning of the interlocutor rather than assuming it is a given.
4. `Viewpoint and Perspectives` - attacks the position from which the viewpoint is given to prove that there are other, equally valid, points of view.
5. `Probing Implications and Consequences` - challenges the logical implications of the argument.
6. `Questions about Questions` - gives the interlocutor the option of viewing the issue from the point of view of the person asking the question.

The Socratic taxonomy, with many extensions, is used in the LSAT[3] multiple-choice tests, mainly as sub-objectives to evaluate the critical thinking abilities. The LSAT questions are intended to measure the complex texts reading and comprehension, the information organization and management, the ability to draw reasonable inferences from the information, and the analysis and evaluation of the reasoning and arguments of others.

1.2. Bloom's Taxonomy

Bloom's taxonomy (Bloom, 1956) is a simple and clear model meant to express qualitatively different levels of thinking. The system contains six levels, arranged from the lowest level of complexity (shallow questions) to the highest (deep questions). Since for these question types there are sets of keywords specific to each type (Dalton, Smith, 1986), the task of automatic QG is more feasible than for the Socratic questions.

1. `Knowledge` - requires recall of information, observation and identifying information in almost the same form it was presented when learned. This type of question is the most frequently used by teachers when assessing student's knowledge. Keywords: *know, who, define, what, where, list, when*.
2. `Comprehension` - asks the interlocutor to take different bits of information and to combine them in order to grasp their meaning and get the answer. Keywords: *describe, use your own words, outline, explain, discuss, compare*.

[2] Critical thinking is the process we use to reflect on, assess and judge the assumptions underlying our own and others ideas and actions.

[3] Law School Admission Test

3. `Application` - tries to take the information known and apply it to a new context with the use of theories, rules and methods learned. Keywords: *apply, manipulate, put to use, employ, dramatize, demonstrate, interpret*, and *choose*.
4. `Analysis` - asks the interlocutor to break down things into their consisting parts, find reasons, causes and reach conclusions (move from whole to the parts). Keywords: *analyze, why, take apart, diagram, draw conclusions, simplify, distinguish*, and *survey*.
5. `Synthesis` - moves from the parts to the whole, as opposed to analysis, and challenges creative thinking in order to get the interlocutor to put together ideas and knowledge from several areas to create new ideas. Keywords: *compose, construct, design, revise, create, formulate, produce*, and *plan*.
6. `Evaluation` - requires judgment on the issue, comparing of ideas, assessing the value of theories and using reasoned arguments to make choices. It does not have a single right answer. Keywords: *judge, assess, value, criticize, compare*.

Since Bloom's taxonomy is more training-oriented, while the Socratic question taxonomy can be mapped better on questions used in real life, the Socratic question types cannot be put in a one-to-one correspondence with the categories from Bloom's taxonomy. Widely used in education and training, the Bloom taxonomy is the starting point for many students' knowledge evaluation developers.

1.3. The taxonomy of Lehnert and Graesser

In the framework of the QUALM questions-answering system, Lehnert (1977) developed a widely used taxonomy of 13 question types, to be further used in the subsequent interpretation and memory search. The taxonomy goes from shallow (`Verification, Disjunctive, Concept completion`), through intermediate (`Feature specification, Quantification`), to deep questions (`Causal antecedent & consequent, Goal orientation, Enablement, Instrumental/procedural, Expectation, Judgmental`). In an analysis of tutoring, Graesser and Person (1992) added other four categories to the previous ones: `Example` (*What is an example of X?*), `Definition` (*What is meant by X?*), `Comparison` (*How is X different from Y?*), and `Interpretation` (*What is the significance of X?*). This taxonomy is the basis for the QA@TREC questions, which are discussed in the next section.

2. Question taxonomies in practical settings

2.1. The LSAT data and taxonomy

The LSAT tests (LSAT, 2009), used for admission in law schools in US, Canada and other countries, measure reading and verbal reasoning skills in a standardized way, without any need of knowledge of the law. The multiple-choice questions (five items with one correct key) are divided into three categories, with other subtypes, each of them based on specially selected and prepared texts.

1. `Reading Comprehension` questions measure the ability to read, understand and reason based on lengthy, complex materials, carefully selected

from humanities, social sciences, biological and physical sciences, and issues related to the law. There are eight reading comprehension question types.

2. `Logical reasoning` questions evaluate the ability to understand, analyze, criticize, and complete a variety of arguments contained in short passages from newspaper articles and editorials, informal discussions and conversations, letters to the editor, speeches, advertisements, as well as articles in the humanities, the social sciences, and the natural sciences. The questions do not presuppose knowledge of the terminology of formal logic. The argument is usually a set of statements in which one is the conclusion which can be drawn from the premises; the argument can be incomplete or it can contain background information. There are seven logical reasoning question types.
3. `Analytical reasoning` questions measure the ability to understand and to draw logical conclusions about the structure of relationships. Starting with a set of statements, rules, or conditions that describe relationships (ordering, grouping, assignment) among entities (things, places, persons, or events), the questions ask to make deductions from them. Since these questions are based on both specially created texts, and specific subtypes, we will not further continue with other classifications of them.

Administered since 1948, currently in four annual sessions, the LSAT tests are manually developed and they are available either as online samples or through books, both retrievable from the LSAC official website (www.lsac.org).

2.2. Taxonomies in Question-Answering evaluation campaigns

Starting with the 8th edition of the Text Retrieval conference – TREC, in 1999, a Question-Answering track was included (Voorhees and Harman, 2000). Using an ad-hoc English text collection (about 2 GB of texts from LA Times, Financial Times, Federal Register and FBIS), a set of questions following the Lehnert taxonomy was manually developed and subsequently provided as input to the participating systems that were supposed to return the answer.

The QA track in TREC 2003 contained list and definition questions in addition to factoid questions (Voorhees, 2004), created based on a news collection. Since 2004 the questions were grouped into series: a question series was focused on a target, and consisted of several Factoid questions, one or two List questions, and exactly one Other type of questions. As it is shown in (Dang et al., 2007), besides news – the AQUAINT-2 corpus, the track used also blogs – the Blog06 corpus. The questions were grouped into 70 series: the order of questions in the series and the type of each question were all explicitly encoded in the test set.

Following the TREC model, a QA track was developed annually in Europe at CLEF – Cross-Language Evaluation Forum since 2003. Based on comparable corpora of news in three languages (Dutch, Italian, Spanish), the DISEQuA corpus was developed, with 450 questions in four languages (including English). With the participating languages growing from year to year, in 2006 (Vallin et al., 2005) there were 7 languages (Bulgarian, Deutsch, English, Finnish, French, Portuguese added to the previous), contributing news corpora from which Factoid and Definition questions were developed.

In 2006 and 2007 the QA@CLEF track included other languages, as well as the List type of questions, and the topic-related questions, with no indications about the topic or the relations between topic-related questions. Another novelty since 2007

(Giampiccolo et al., 2008) was the use of the Wikipedia text collections, available for all languages involved in the campaign. For the 2009 campaign the corpus used in the ResPubliQA track at CLEF is the Acquis Communautaire (Steinberger et al., 2006) – a parallel collection of the European laws, available in 22 languages. The taxonomy of questions is changed (Factoid, Definition, Purpose, Reason, Procedure) and the question set consists of unrelated questions.

Most of the data (text corpora and question collections) are freely available through the TREC-QA (trec.nist.gov/data/qa.html) and QA@CLEF (nlp.uned.es/clef-qa/) websites. A mapping between the CLEF and TREC QA tracks is easily achievable because the question types and subtypes have the same or synonym names.

3. Conclusions and future work

Since adequately chosen question taxonomies are an important component in a question-generation system, the paper gives a detailed overview of the main theoretical and applied taxonomies, indicating also important data available for their use. We choose these taxonomies because they are among the mostly used in the research community, they cover various domains, and, more important, the LSAT, TREC and CLEF questions can be used in a multilingual setting for a QG campaign and their data are freely available (text corpora, questions, and their answers).

These taxonomies will be used in the development of a QG prototype, which will be also evaluated based on the available data.

References

[1] B.S. Bloom, *'Taxonomy of Educational Objectives: Handbook 1: Cognitive Domain*. Addison Wesley Publishing Company, 1956.

[2] J. Dalton, D. Smith, *Extending Children's Special Abilities – Strategies for primary classrooms*, Melbourne : Curriculum Branch, Schools Division, 1986.

[3] H.T. Dang, D. Kelly, J. Lin, Overview of the TREC 2007 Question Answering Track, in E. Voorhees (ed.) *The 16th Text REtrieval Conference Proceedings*, NIST Special Publication SP 500-274, 2007.

[4] D. Giampiccolo, P. Forner, A. Peñas, C. Ayache, C. Forăscu, V. Jijkoun, P. Osenova, P. Rocha, B. Sacaleanu, R. Sutcliffe, Overview of the CLEF 2007 Multilingual QA Track. In Carol Peters et all. (eds.) *Advances in Multilingual and Multimodal Information Retrieval*, LNCS 5152, Springer, 2008.

[5] A. Graesser, N. Person, and J. Huber, Mechanisms that generate questions. In T. Lauer, E. Peacock, and A. Graesser (Eds), *Questions and information systems*. Earlbaum, Hillsdale, 1992.

[6] W. Lehnert, *The Process of Question Answering*. ARDA Research Report, May 1977.

[7] LSAT: *LSAT Information Book*, Law School Admission Council, 2009; available at http://www.lsac.org/

[8] R Paul, *Critical Thinking: How to Prepare Students for a Rapidly Changing World*. Foundation for Critical Thinking, 1993.

[9] V. Rus, and A.C. Graeser (eds.), *The Question Generation Shared Task And Evaluation Challenge*, Workshop Report, University of Memphis, ISBN:978-0-615-27428-7. February 2009.

[10] A. Vallin, D. Giampiccolo, L. Aunimo, C. Ayache, P. Osenova, A. Peñas, M. de Rijke, B. Sacaleanu, D. Santos, R. Sutcliffe, Overview of the CLEF 2005 Multilingual Question Answering Track. In *Accessing Multilingual Information Repositories*, LNCS 4022, Springer, 2006.

[11] R. Steinberger, B. Pouliquen, A. Widiger, C. Ignat, T. Erjavec, D. Tufiş, D. Varga, The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC'2006)*. Genoa, Italy, 24-26 May 2006.

[12] E. Voorhees, D. Harman, *Overview of the Eighth Text REtrieval Conference (TREC-8)*, NIST Special Publication 500-246, 2000.

[13] E. Voorhees, Overview of the TREC 2003 Question Answering Track. In *Proceedings of the Twelfth Text REtrieval Conference (TREC 2003*), NIST Special Publication 500-255, 2004.

Ranking Automatically Generated Questions as a Shared Task

Michael HEILMAN, Noah A. SMITH
Language Technologies Institute, Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA, 15213, USA

Abstract. We propose a shared task for question generation: the ranking of reading comprehension questions about Wikipedia articles generated by a base overgenerating system. This task focuses on domain-general issues in question generation and invites a variety of approaches, and also permits semi-automatic evaluation. We describe an initial system we developed for this task, and an annotation scheme used in the development and evaluation of our system.

Keywords. question generation, human language technologies

Introduction

Questions are an important component of many educational interactions, from one-on-one tutoring sessions to large-scale assessments. Thus, the automation of generating questions would enable the efficient development of flexible and adaptive instructional technologies. For example, a teacher might use a question generation (QG) system to quickly create assessments for daily reading assignments. Or, he or she might use an automated system to generate comprehension questions about a text he or she found on the web, facilitating the use of practice reading texts that closely match students' interests.

In addition, the utility of automated QG extends beyond educational technology into fields such as web search and dialogue systems. A more extensive discussion of QG and its applications can be found in the report from the NSF-sponsored Workshop on the Question Generation Shared Task and Evaluation Challenge [1].

Each application of QG will have a slightly different set of criteria for evaluating the quality of the questions produced. For instance, in a tutoring application, the quality of questions may depend on the state of a learner model. On the other hand, in a dialogue system, quality may be determined by the efficiency and success of a commercial transaction. Nonetheless, we claim that certain characteristics of questions are domain-general, having relevance in educational applications as well as other domains. This set of general characteristics addresses issues such as grammaticality, vagueness, the use of the appropriate WH-word, and the presence of formatting errors in the output.

General characteristics of questions such as grammaticality are often quite difficult to evaluate automatically. As such, manual judgments of question quality are informative, and perhaps necessary, for effective QG evaluations. Researchers in related fields have developed semi-automatic measures of quality for their respective tasks (e.g., in machine translation [2] and text summarization [3]). While these metrics require some initial human input (e.g., reference translations, human summaries), we note that their value derives from the fact that they facilitate efficient evaluations by making the human input reusable.

In this paper, we contribute to the study of QG by proposing a shared task, the ranking of reading comprehension questions about Wikipedia articles, which focuses on application-general QG issues (§1). We describe a system we developed for this task (§2), and an initial annotation scheme and process (§3) that we used to evaluate the system. We also describe relevant previous research (§4), and conclude with a general discussion (§5).

1. Proposed Task: Ranking Questions about Wikipedia Articles

We propose as a shared QG task the ranking of automatically generated reading comprehension questions about Wikipedia articles. At least in the first version of the task, the questions would be about literal information in the articles, rather than dealing with more complex questions involving inference and prior knowledge. Additionally, the task would assume that the student has only minimal prior knowledge of the topic of the text.

It is important to emphasize that the task would not be the *generation* of questions, but rather the *ranking* of questions generated by a base system that overgenerates (i.e., produces a large amount of output in order to increase the chance of including more high-quality items, perhaps at the expense of including a higher percentage of low-quality items). For example, from the sentence *Francium was discovered by Marguerite Perey in France in 1939*,[1] an overgenerating system might produce the following questions:

- *Where was francium discovered by Marguerite Perey in 1939?*
- *When was francium discovered by Marguerite Perey in France?*
- *Was francium discovered by Marguerite Perey in France in 1939?*
- *By what was francium discovered in France in 1939?*

A system participating in the task would take as input a large number, possibly in the hundreds or more, of unranked literal comprehension questions such as the above, along with metadata formally describing how each question was produced from the source article. This metadata might include the original source sentence, the location of that sentence, the linguistics transformations performed by the overgenerator, etc. The system would then produce as output a single ranking of the input questions by their predicted levels of acceptability. For instance, for the example just presented, a system might identify the last question as unacceptable (because it uses the inappropriate WH-word) and therefore penalize it in the ranking. Acceptability would be defined for evaluation purposes as

[1] From "Francium." *Wikipedia: The Free Encyclopedia*. Retrieved Dec. 16, 2008.

the satisfaction of *all* of a set of domain-general criteria (such as those described in §3).

Researchers have found overgenerate-and-rank approaches to be successful in various domains, including natural language generation [4,5] and syntactic parsing [6]. Initially, we propose that the overgenerating system be based on lexical and syntactic transformations of sentences from the input text. In order to transform declarative sentences into questions, the overgenerator would perform linguistic transformations such as subject-auxiliary inversion, WH-movement, and replacement of answer phrases with appropriate WH-words (i.e., *who*, *what*, *where*, etc.).

The relative simplicity of this ranking task allows researchers to focus on domain-general issues that are relevant to QG in various application domains. In contrast, a QG task in the context of a tutoring system might focus too much on the interactions of question quality with students' prior knowledge for its results to be relevant to a QG researcher working on a dialogue system for commercial transactions. On the other hand, a QG task in the context of a dialogue system for flight reservations might focus too much on the step-by-step communicative process for its results to be relevant to a QG researcher working on tutoring. The issues that the ranking task focuses on, however, are important in many domains.

Of course, participants' ranking systems would to some extent depend on the specific evaluation criteria used for evaluating this task. However, since the criteria are intended to be domain-general, they would likely constitute a subset of the criteria for many specific QG applications. Therefore, to build a ranking system for a specific application, a participant could *extend* rather than replace the ranking system—for example, by adding additional features for a machine learning algorithm to consider when learning to rank.

Another benefit of the ranking task is that human judgments could be reused, making evaluation a semi-automatic process as in machine translation and summarization. Only a single round of human annotation of the questions from the base overgenerating system would be necessary. Once this single set of ratings is acquired, new question-ranking systems could be evaluated automatically by examining the original human judgments of the questions that they rank the highest (e.g., in the top-10).

In subsequent versions of the task, the overgenerator might be revised, replaced, or augmented to address more complex semantic and discourse phenomena. Altering the overgenerator would also deal with the potential problem of focusing too much on the idiosyncrasies of a particular overgenerating approach. An additional shared task for building overgenerating systems also seems possible. For such a task, the overgenerating systems might be evaluated by combining them with various question ranking systems developed for the ranking task.

2. Description of an Implemented System for this Task

We have developed an initial system that overgenerates and ranks questions about English Wikipedia and Simple English Wikipedia articles. The system works in three stages, the first two of which correspond to the overgenerating system de-

scribed previously, and the third of which corresponds to the ranking system that participants in the shared task would build.

In stage 1, sentences from the text are transformed into declarative sentences by optionally altering or transforming lexical items, syntactic structure, and semantics. Many existing NLP transformations may be exploited in this stage, including extractive summarization, sentence compression, sentence splitting, sentence fusion, paraphrase generation, textual entailment, and lexical semantics for word substitution. In our system, we have included a small set of transformations based on previous work in headline generation [7] and summarization [8], such as the removal of appositives and adverbial modifiers.

In stage 2, the derived declarative sentence is turned into a question by executing a set of well-defined syntactic transformations (WH-movement, subject-auxiliary inversion, etc.). The system explicitly encodes well-studied linguistic constraints on WH-movement such as noun phrase island constraints [9]. The transformation rules were implemented by automatically parsing the input into phrase structure trees with the Stanford Parser [10] and using hand-written rules in the Tregex and Tsurgeon tree searching and tree manipulation languages [11] to modify those trees. Note that both the automatic statistical parsing and manually defined transformation operations introduce errors, which further motivate the introduction of a ranking component to identify situations in which errors are likely to have occurred.

In stage 3, which corresponds to the proposed ranking task, the questions are scored and ranked using a statistical classifier based on various features of the questions from stage 2. The system will be described in detail in a forthcoming technical report.

3. Annotation for Evaluation Questions According to General Criteria

In this section, we describe the annotation scheme we used to evaluate our rating system. The annotation process was imperfect, both in the categorization of question deficiencies and the execution of the manual annotation process, but a few key alterations would lead to a more effective and reliable annotation scheme.

We defined a set of question deficiencies, listed and described in Table 1, which focus on characteristics of questions that are reasonably domain- and task- independent, such as grammaticality and appropriate specificity. Since these aspects of questions are not mutually exclusive, the annotator makes a binary decision for each category as to whether a given question is acceptable or not according to that factor. For example, a question might be both vague and ungrammatical, and, in such a case, an annotator would mark both deficiencies. Note that the "Acceptable" category is for questions that are not deficient according to any of the other factors, and *is* therefore mutually exclusive with the others.

The set of questions used in our evaluation test set included 644 questions generated from 8 articles. Four articles, two each, came from the English Wikipedia and the Simple English Wikipedia. The others were two pairs of articles from the Wall Street Journal section of the Penn Treebank, with one of each pair including automatic syntactic parses from the Stanford Parser [10], and the other including

the human-annotated gold-standard syntactic parses from the Treebank. We used the Penn Treebank texts to explore the effects of parser output quality on the generated questions, but we will not go into detail on those results here. Three persons annotated the questions from each article.

Inter-rater agreement, measured by Fleiss's κ, was fairly low across the categories. For the most important distinction, between questions that are acceptable and those that possess any deficiency, the κ value was .42, corresponding to "moderate agreement" [12]. From observations of the data and comments by annotators, it appears that a few alterations to the scheme would improve agreement. For instance, the categories of "Ungrammatical" and "Does not make sense" might be merged. While the first focuses on syntactic errors and the second on semantic errors, this distinction was not readily apparent to annotators. Additionally, these two types of deficiency can often be attributed to similar causes, such as syntactic parsing errors. The infrequent "Missing Answer" category could also be merged into "Other". Of course, it would be possible to merge all of the categories and simply rate questions as good or bad. However, the identification of the causes of erroneous output likely has value, both from a scientific point of view, for understanding which aspects of QG are most challenging and interesting, as well as from an engineering point of view, in that a statistical ranking model may be able to leverage this information when ranking questions.

Also, some of the categories could benefit from more detailed descriptions, and though we provided a few examples with each category, more positive and negative examples for each would likely increase agreement. The "Obvious Answer" category was particularly problematic in this regard.

We also note that many of the guidelines and methods for training annnotators and improving annotation schemes (cf. [13]) will likely be useful for QG evaluation.

In addition to the set of categories used to annotate questions, improving the *process* of annotation–i.e., annotator training, spot-checking, and redundancy–would likely lead to higher agreement. Three novice annotators rated each article in the test set for our experiments, and because of the relatively low agreement, questions were judged to be "acceptable" only when no annotator annotated it with any deficiency. Other annotation processes might involve either a smaller number of extensively trained annotators, or a much larger number of novice annotators rating questions redundantly through a facility such as Amazon.com's Mechanical Turk (cf. [14]).

For the evaluation metric for this task, we propose using the percentage of questions labeled "Acceptable" in the top N questions, or precision-at-N. This metric is appropriate because a typical user would likely consider only a limited number of questions. Precision at a single N value (e.g., $N = 10$) would serve as the primary metric, and precision at other N ranging from 1 to 30 would provide additional detail.

4. Prior Work

In this section, we describe some of the prior work relevant to the proposed shared task on ranking questions.

Deficiency	Description	%	κ
Ungrammatical	The question does not appear to be a valid English sentence.	36.7	.29
Does not make sense	The question is grammatical but indecipherable. (e.g., *Who was the investment?*)	39.5	.29
Vague	The question is too vague to know exactly what it is asking about, even after reading the article (e.g., *What did Lincoln do?*).	40.2	.40
Obvious answer	The correct answer would be obvious even to someone who has not read the article (e.g., the answer is obviously the subject of the article, or the answer is clearly *yes*).	14.7	.13
Missing answer	The answer to the question is not in the article.	5.1	.14
Wrong WH word	The question would be acceptable if the WH phrase were different (e.g., *in what* versus *where*). WH phrases include *who*, *what*, *where*, *when*, *how*, *why*, *how much*, *what kind of*, etc.	12.2	.40
Formatting	There are minor formatting errors (e.g., with respect to capitalization, punctuation)	18.3	.50
Other	There are other errors in the question that are not covered by any of the categories	10.8	.03
Acceptable	None of the above deficiencies were present.	12.8	.42

Table 1. The categories used in our evaluation to describe the possible deficiencies a generated question may exhibit. On the right are, for each category, the percentages of questions that at least one annotator annotated with that category as well as the Fleiss's κ value for inter-annotator agreement.

Overgenerate-and-rank and reranking approaches have been applied successfully in various domains. In particular, Walker et al. [4] rank potential output from a natural language generation system based on features that correlate with human judgments of quality. Langkilde and Knight [5] also rank output of a generation system, but using corpus evidence rather than human judgments. Similar ranking approaches have been employed for natural language processing tasks. Many state-of-the-art approaches to syntactic parsing currently re-rank their output based on features that cannot be easily incorporated into the pre-ranking component [6].

Most previous evaluations of QG systems have been domain- or task-specific and relatively small-scale (e.g., [16], [17]). Humans rated the output of a particular system. As such, if further developments were made in QG, even for the same task, another round of human judgments would be required.

Evaluations for similar domains are also informative. In summarization [3] and machine translation [2], initial human input in the form of reference translations and summaries have led to semi-automatic metrics. For a given input sentence or document, these metrics compute the surface similarity of system output, of which there are infinitely many possibilities, to output produced by a human. However, it is unclear whether such an approach could be directly adapted for QG. Furthermore, criticisms of metrics based on surface similarity are well-known [18].

Human evaluations are often used in such tasks, particularly for final evaluations rather than ongoing development. The rating schemes often focus on aspects of the output that are relevant to QG, such as grammaticality (e.g., [19]).

In tasks such as recognizing textual entailment [20], paraphrase identification [21], and question answering [22], semi-automatic measures have also been developed based on initial human input. Unlike the above tasks, the output in these cases is a classification decision (e.g., paraphrase or not, correct or incorrect answer). This makes the assessment of correctness of system output straightforward.

5. Discussion

An ideal QG system would be broad-domain (e.g., would work for tutoring in science or history, and for elementary school or college), scalable to large datasets, and generate questions involving deep inference about the source material. Such a solution is not likely to be feasible with current semantic representations and automatic natural language processing technologies.

The feasible approaches to QG fall along a range with respect to their domain-generality, scalability, and the depth of the linguistic and cognitive issues that they address. At one end of this range are approaches that focus on particular narrow-domain QG applications. While such approaches are able to consider deep aspects of linguistics (e.g., semantics, discourse processing) and human knowledge in their specific domains, they make use of resources (e.g., complex ontologies) that may be expensive and not useful across systems.

At the other end of this range are broad-domain, scalable approaches—such as the ranking task proposed here—that focus on more general (though perhaps more shallow) aspects of language, questions, and human knowledge. Focusing on such issues would spur innovation on generalizable and scalable techniques that would be relevant to a variety of specific QG applications. Even very narrow-domain applications would likely benefit because they must still address issues such as grammaticality and vagueness.

In conclusion, the domain-generality of the proposed task and the existence of a straightforward automated evaluation method, as discussed in §1, are the primary benefits that make it likely to succeed in furthering research on QG.

Acknowledgements

We thank the anonymous reviewers for their helpful comments. This work was supported by an NSF Graduate Research Fellowship, Institute of Education Sciences grant R305B040063, and DARPA grant NBCH-1080004.

References

[1] V. Rus and A. Graesser, Eds. *The Question Generation Shared Task and Evaluation Challenge*, ISBN:978-0-615-27428-7 (2009).

[2] K. Papineni and S. Roukos and T. Ward and W.-J. Zhu, BLEU: A method for automatic evaluation of machine translation, *Proc. of ACL* (2002).
[3] C. Lin, ROUGE: A package for automatic evaluation of summaries, *Proc. of Workshop on Text Summarization* (2004).
[4] M. A. Walker and O. Rambow and M. Rogati, SPoT: A trainable sentence planner, *Proc. of NAACL* (2001), 1–8.
[5] I. Langkilde and K. Knight, Generation that exploits corpus-based statistical knowledge, *Proc. of ACL* (1998).
[6] M. Collins, Discriminative reranking for natural language parsing, *Proc. of ICML* (2000).
[7] B. Dorr and D. Zajic, Hedge Trimmer: A parse-and-trim approach to headline generation, *Proc. of Workshop on Automatic Summarization* (2003).
[8] K. Toutanova and C. Brockett and M. Gamon and J. Jagarlamudi and H. Suzuki and L. Vanderwende, The PYTHY summarization system: Microsoft Research at DUC 2007, *Proc. of DUC* (2007).
[9] N. Chomsky, On wh-movement, In P. W. Culicover and T. Wasow and A. Akmajian (Eds.), *Formal Syntax*, New York: Academic Press (1977).
[10] D. Klein and C. D. Manning, Fast exact inference with a factored model for natural language parsing, *Advances in NIPS 15* (2003).
[11] R. Levy and G. Andrew, Tregex and Tsurgeon: Tools for querying and manipulating tree data structures, *Proc. of LREC* (2006).
[12] J. R. Landis and G. G. Koch, The measurement of observer agreement for categorical data, *Biometrics* **33** (1977) 159–174.
[13] K. Krippendorff, *Content Analysis: An Introduction to Its Methodology, 2nd Edition*, Sage (2004).
[14] R. Snow, B. O'Connor, D. Jurafsky and A. Ng, Cheap and fast – but is it good? Evaluating non-expert annotations for natural language tasks, *Proc. of EMNLP* (2008).
[15] L. von Ahn and L. Dabbish, Labeling images with a computer game, *Proc. of the SIGCHI conference on Human factors in computing systems* (2004).
[16] R. Mitkov and L. A. Ha and N. Karamanis, A computer-aided environment for generating multiple-choice test items, *Natural Language Engineering* **12** (2006), 177–194.
[17] H. Kunichika and T. Katayama and T. Hirashima and A. Takeuchi, Automated question generation methods for intelligent English learning systems and its evaluation, *Proc. of ICCE* (2004).
[18] C. Callison-Burch and M. Osborne, Re-evaluating the role of BLEU in machine translation research, *Proc. of EACL* (2006), 249–256.
[19] K. Knight and D. Marcu, Statistics-based summarization - step one: sentence compression, *Proc. of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence* (2000).
[20] O. Glickman and I. Dagan and M. Koppel, A probabilistic classification approach for lexical textual entailment, *Proc. of AAAI-05* (2005).
[21] W. B. Dolan and C. Brockett, Automatically constructing a corpus of sentential paraphrases, *Proc. of IWP2005* (2005).
[22] E. M. Voorhees, Overview of the TREC 2003 question answering track, *Proc. of TREC 2003* (2004).

Generation of exercises within the PERLEA project

Stéphanie Jean-Daubias, Marie Lefevre, Nathalie Guin
Université de Lyon, France
Université Lyon 1 – LIRIS, CNRS, UMR5205
43 bd du 11 novembre 1918, 69622 Villeurbanne Cedex, France
{Stephanie.Jean-Daubias, Marie.Lefevre, Nathalie.Guin}@liris.univ-lyon1.fr

Abstract: The research we have carried out relates to the personalization of learning thanks to the exploitation of learners profiles through the PERLEA project. We are aiming at designing a module managing the generation of personalized activities. For this purpose, we suggested a typology of pencil and paper exercises that can be given to a learner, as well as the architecture of generators allowing the creation of all of these exercises. We also implemented and tested our proposition in a module helping the teacher to propose exercises suited to his students' knowledge.

Keywords. Interactive Learning Environments (ILE), personalization, generation of exercises, architecture, genericity.

1. Introduction

Personalization of learning is one of the major issues of Technology Enhanced Learning. Personalization relies in particular on using learners profiles to gather information about the learners, thus allowing to describe their knowledge, skills, perceptions and/or behaviors. These data are collected or deduced from one or several pedagogical activities, computerized or not [6].

Our approach consists in helping the teacher proposing to learners personalized pedagogical activities suited to their knowledge and gaps shown in their profiles, and suited to the teacher's needs and to the pedagogical context, expressed in what we name pedagogical strategies. To personalize pedagogical activities offered to the learner based on their profile, we can either use knowledge-based systems to generate the pedagogical activities best-suited to the profile, or provide the teachers with tools allowing them to perform this task themselves. We aim at linking these two options.

In this paper[1] we focus on the exercises generation part of our research. To build the Adapte module, we proposed a typology of exercises that can be given to a learner, together with the architecture of eight generators able to create all of these exercises. We detail these two aspects before moving on to their implementation and validation.

The PERLEA project aims at improving the integration of ILEs in education by building bridges between the use of ILEs and teachers' everyday practices. To do so, we are interested, in a generic way, in learners profiles and their a posteriori use for the

[1] A long version of this paper is available in the research report RR-LIRIS-2009-016.

management of learners and the personalization of learning [6]. Hence we aim at developing an environment that would permit teachers to manipulate existing profiles. This environment consists of two phases: the integration of existing profiles (based on PMDL, the profiles modeling language that we proposed to unify external learners profiles to permit their reuse, either pencil and paper or software ones [7]) and the management of thus unified profiles. The second phase of the environment proposes rich uses of the unified profiles. One of such uses is accomplished by the Adapte module, which offers to learners activities adapted to their profiles. These activities may be pencil and paper activity generated by the system (worksheets to be printed) or computerized activities to be done in an external ILE. In the case of pencil and paper activities, Adapte generates worksheets matching the profile of each learner, according to teacher's pedagogical goals. To achieve this, it creates tailor-made exercises to be included in the sheets and determines the size of the worksheets themselves. It also provides the teacher with the answers to the exercises contained in the sheets. In the case of computerized activities, Adapte sets personalized sessions on external ILEs according to the learners profile. For this, it uses ILE exercises generators or chooses exercises in the ILE database. It also computes the number of exercises, in which order they appear and the duration of the session.

2. Generation of pencil and paper activities

2.1. Typology of exercises

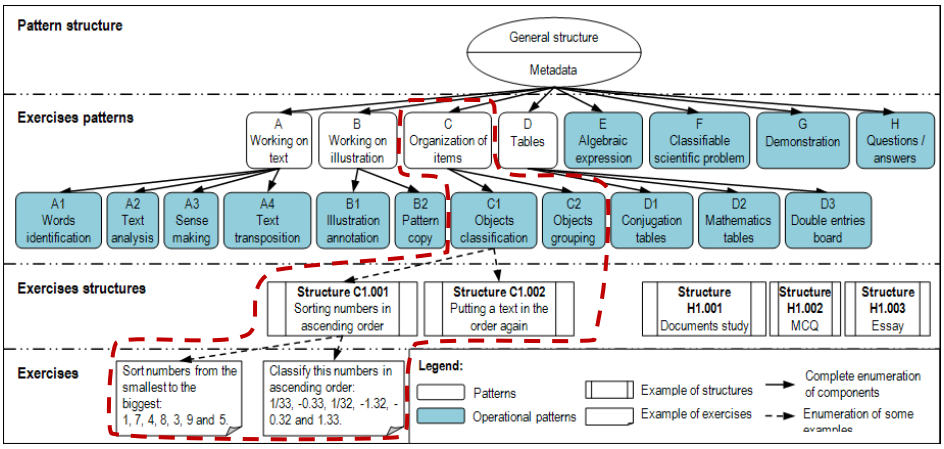

Figure 1: Typology of pencil and paper exercises

By studying curricula published in the official texts of the French Ministry of Education, and subsequently working with teachers in elementary schools, as partners in the PERLEA project, we have identified fifteen types of exercises that can be proposed to a learner, taking into account all subjects and levels. The identified typology of exercises is presented in Figure 1. Our typology contains eight exercises patterns (see A to H in Figure 1), some of which can be split into several operational patterns. An *exercises pattern* (e.g. C - Organization of items, in Figure 1) defines a category of exercises generated with the same exercises generator. An *operational*

pattern (e.g. C1 - Classifying objects) specifies a subset of exercises generated through the pattern generator (here C), but with particular generation constraints. Our typology contains fifteen operational patterns defining fifteen types of exercises. The generic structure of these patterns and the set of metadata common to all patterns are defined in a patterns structure. From there, creating an *exercises structure* consists in associating an operational pattern with generation constraints. Creating an *exercise* consists in assigning to the parameters of the exercises structure values that satisfy these constraints. Thus created exercises are composed of elements of wording and elements of answer proposed to the learner, as well as the solution to the teacher.

2.2. Generation of exercises

2.2.1. What type of generator for Adapte?

Existing exercises generators can be classify into three categories. Fully *automatic generators* generate exercises without any intervention of the user [1] [2]. They permit to quickly create a large number of exercises, but are not customizable by teachers who can neither adapt them to their work habits, nor to their students' specificities. On the opposite, *manual generators* (authoring tools) guide the user in the exercises design [3]. They give the teacher complete freedom. But, he must fully define the exercises and their solutions, which is a tedious task that restraints the use of such systems. Half-way between these two types, *semi-automatic* generators can construct the terms of exercises themselves, but allow the user to intervene in the creation process by specifying a set of constraints on the exercises [4] [8]. Semi-automatic generators have the same strengths as automatic generators (quickly generating a large number of exercises) and provide a solution to their lack of flexibility: teachers can tune the parameters of generated exercises.

The most suitable approach is for us to incorporate semi-automatic generators. We use this approach, which relies on the teacher to provide the knowledge bases for the semi-automatic generators, in cases where state of the art semi-automatic generators seem unrealistic in our generic context. We then studied the possibility for each Adapte exercises pattern to use existing generators. If we except the F-type exercises of Figure 1, with the generators which were available to us, the teacher has either to key in the exercises completely or he cannot influence at all the creation process. Using such types of generators would have prevented us to propose a random option to teachers in the generation of their exercises. For categorized scientific problems (F-type in Figure 1), we integrated into Adapte, GenAMBRE, the generator of AMBRE-Teacher [4] [8], implemented to create arithmetic word problems in the AMBRE-add ILE. By providing the necessary knowledge bases, this generator could be used in a generic way and thus provide exercises on problems of combinatorial analysis, thermodynamics, etc.

2.2.2. Architecture of semi-automatic generators

To each exercises pattern presented in Figure 1 corresponds a generator that creates exercises for the learner and answers for the teacher. An answer will be either defined by the generator if possible, or keyed in by the teacher. If some constraints are not specified by the teacher, they will be specified by the system. Moreover, at the time when an exercise is generated, the exercises structure may contain constraints of re-generation preventing the same exercise to be generated again for the same exercises structure. All generators proposed for Adapte comply with a generic architecture (see

Figure 2). The knowledge of the generators is provided partly by the designer of the system, and partly by the teachers who thus complete gradually the knowledge bases.

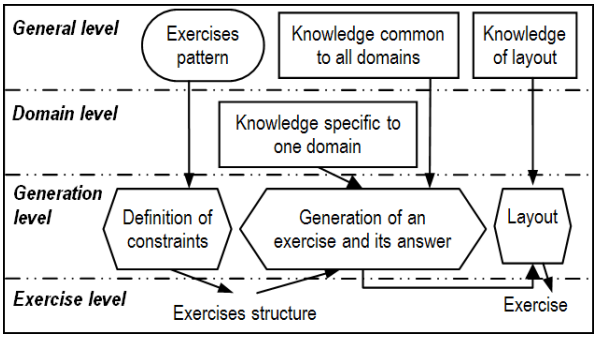

Figure 2: Generic architecture of exercises generators

Thanks to this generic architecture of exercises generators, we can specify four levels. The *general level* contains the knowledge common to all domains for which we want to generate an exercise, e.g. the knowledge required to write a statement in natural language. The *domain level* contains the knowledge specific to the domain, e.g. the knowledge of calculation. The *generation level* contains the specific processes to create an exercise: definition of constraints on an exercises pattern saved in an exercises structure; instantiation of this structure to generate an exercise and its answer; layout enabling to provide exercises with a uniform presentation. Finally, the *exercise level* contains all the documents for the created exercise, including the exercises structure and its instantiation (wording of the exercise and its answer). We specialized the generic architecture alike to define the exercises generators associated with the exercises patterns except for the "Demonstration" pattern (G in Figure 1).

These architectures are implemented in the Adapte module. When a teacher wants to create an exercise, he has first to choose the corresponding type of exercises. From this operational pattern, the system presents the teacher with an interface enabling him to define the constraints of exercise generation depending on his pedagogical goal. For example, for the conjugation operational pattern, the teacher chooses a language and can specify the tense, persons, types of verbs (regular or irregular for English language) and/or verbs, and the number of verbs to be proposed to the learners. All these constraints are saved in an exercises structure, described with metadata to facilitate its reuse. The system generates the exercises contained in the personalized worksheets from this exercises structure. Thus it generates different exercises from the same exercises structure.

3. Conclusion

We established an approach of personalization of learning helping teachers to propose pedagogical activities suited to learners' knowledge and to teachers' needs. In this framework, we focused here on the generation of pencil and paper exercises.

First, we presented our typology of exercises that can be given to a learner from primary to high school. This typology includes fifteen types of exercises. We defined it with the primary school teachers associated to the project and test its scope with

secondary teachers. We observed each of the exercises they use for their French, English, mathematics, biology, history and geography classes. All the exercises used were in our typology. Now, we have to work with experts in educational science to completely validate our typology, both in its genericity and its completeness.

We then proposed a generic architecture of exercises generators and set the architectures of the eight exercises generators that we considered necessary to create the exercises of our typology. These generic architectures can be used to develop exercises generators whatever context they are meant to be used in. If these architectures facilitate the setting up of generators in new domains of application, there is left to do a considerable work of instantiation of knowledge bases for a new domain. We were able to test the genericity of these generators by implementing some of them in varied domains (for example we have implemented the tables generator to propose conjugation exercises but also multiplication or addition ones).

We also developed Adapte in partnership with teachers according to differentiated design [5] and submitted it to few teachers. Every feedback validates the software: it is usable and permits teachers to define the constraints allowing to generate exercises matching their needs (expressed in their constraints) and their learners' knowledge (due to Adapte functionalities not presented in this paper [9]). We must now make further evaluations involving all concerned modules of the PERLEA project environment [6], and ranging from the description of a learners profiles structure by the teacher [7] to the effective use of personalized activities by learners [9].

References

[1] Bouhineau, D., Bronner, A., Chaachoua, H. and Nicaud, J.-F. (2008). Helping Teachers Generate Exercises with Random Coefficients. International Journal of Continuing Engineering Education and Life-Long Learning, vol 18-5/6, 520-533.

[2] Burton, R. R. (1982). Diagnosing bugs in a simple procedural skill. Intelligent Tutoring Systems. London, Academic Press, 157-184.

[3] David, J.-P., Cogne, A. and Dutel, A. (1996). Hypermedia exercises prototyping and modelising. Computer Aided Learning and Instruction in Science and Engineering. S. B. Heidelberg: 252-260.

[4] Guin-Duclosson, N. (1999). SYRCLAD : une architecture de résolveurs de problèmes permettant d'expliciter des connaissances de classification, reformulation et résolution. Revue d'Intelligence Artificielle, vol. 13-2, éditions Hermès, 225-282.

[5] Jean-Daubias, S. (2009). Differentiated design: a design method for ILE. Research report RR-LIRIS-2009-015.

[6] Jean-Daubias, S. and Eyssautier-Bavay, C. (2005). An environment helping teachers to track students' competencies, Workshop Learner Modelling for Reflection, Artificial Intelligence in Education (AIED'2005), Netherlands, 19-23.

[7] Jean-Daubias, S., Eyssautier-Bavay, C. and Lefevre, M. (2009). Harmonization of heterogeneous learners profiles, Research report RR-LIRIS-2009-013.

[8] Jean-Daubias, S. and Guin, N. (2009). AMBRE-teacher: a module helping teachers to generate problems, 2nd Workshop on Question Generation, Artificial Intelligence in Education (AIED'2009), Great Britain.

[9] Lefevre, M., Cordier, A., Jean-Daubias, S. and Guin., N. (2009). A Teacher-dedicated Tool Supporting Personalization of Activities, World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED-MEDIA 2009), Hawaii.

AMBRE-teacher: a module helping teachers to generate problems

Stéphanie Jean-Daubias, Nathalie Guin
Université de Lyon, France
Université Lyon 1 – LIRIS, CNRS, UMR5205
43 bd du 11 novembre 1918, 69622 Villeurbanne Cedex, France
{Stephanie.Jean-Daubias, Nathalie.Guin}@liris.univ-lyon1.fr

Abstract. If teachers use few ILEs (Interactive Learning Environments) with their students, it may be because they don't have the opportunity to act upon the available environments. To permit the adoption of ILEs by teachers, these systems must be adaptable to the learning context and to the teacher's pedagogical approach. To achieve this we propose a module dedicated to the teacher in the context of the AMBRE project. This module, AMBRE-teacher, allows the user to configure the ILE he wants to use. We identified the functionalities necessary to adapt an AMBRE ILE. We notably designed a knowledge based system allowing the teacher to generate the problems he wants to propose to his students in the ILE. Our approach is implemented for AMBRE-add, an ILE proposing word additive problems in elementary school.

Keywords. adaptation of ILE, role of teacher, problem generation, knowledge.

1. Introduction

The AMBRE project is a multidisciplinary study in computer science, cognitive sciences, mathematical didactics and educational science, which aims at designing learning environments for the acquisition of a specific method in problem solving. In each application domain, a method is based on the classification by the learner of problems and solving tools. We propose, to help the learner acquire such methods, to use Case-Based Reasoning, a paradigm developed in Artificial Intelligence and inspired by research in psychology on reasoning by analogy. After the evaluation of a first prototype for the numbering problems domain (final scientific year level, 18 year-old students), we implemented and tested a complete system for additive word problems solving which are studied in primary school: AMBRE-add [7].

In order to facilitate the integration of AMBRE in education, we conceived a module dedicated to teachers, AMBRE-teacher. In this paper[1], after presenting the functionalities of this module, we describe the functionality dedicated to problem generation and present the architecture of the knowledge based system allowing to implement it.

[1] A long version of this paper is available in the research report RR-LIRIS-2009-017.

2. AMBRE: "which role for the teacher?"

Most ILEs (Interactive Learning Environments) focus on the computer/learner couple, thus often masking the important role of teachers in these environments [9]. The teacher can first directly participate at the design of the ILE as designer or design partner. He can also be an ILE designer as user of an authoring tool. In addition, the teacher usually takes the role of prescriber by choosing the system that he will propose to his students. The teacher is also sometimes a secondary user of the ILE used by his students when he has to tune it. Lastly and more rarely, the teacher is the main user of a system specifically designed to help him in his teaching task.

In the AMBRE project, if several teachers took part as **design partners** in the framework of differentiated design [5], teachers' major role is **main user** of a module dedicated exclusively to them. Thus they explicitly take their role of **secondary user** of the AMBRE learner module through the proposed environment, by adapting and defining the parameters of the learner environment. Finally, teachers keep their role of **prescribers**, by choosing the ILE used in their classroom.

For AMBRE, we wish to propose to the teacher an environment designed for him and allowing him to integrate and adapt an AMBRE ILE to his approach, his pedagogical strategies and also to the context of learning. To achieve this, AMBRE-teacher comprises five functionalities. The **configuration of the learner environment** consists in the customization by the teacher of the learner environment interface and of the elaboration of students lists. Even if the learner environment includes a set of predefined exercises, AMBRE-teacher contains a module for **problem generation** enabling teachers to propose to their students problems for which they define the characteristics. Teachers can also **create learning sequences** (sets of problems to solve in one or several sessions of software use) by using the problems they created and by tuning the functioning of the learning environment in terms of help and of diagnosis for the exercises of the sequence. AMBRE-teacher also allows to **distribute the work** to the learners, that is to say to associate the whole group or each learner with one or several sequences. Finally, to allow teachers to entirely adapt the generated problems to the context, AMBRE-teacher can **manage surface features** to be used in the exercises: theme, objects and associated characters, actions, etc.

3. A problem generator for AMBRE

We have chosen a semi-automatic generators approach [4] [8] [3], which builds the wording of the problems, but let the user intervene in the creation process. Actually, on the one hand, automatic generators [1] [6], permitting no interaction with users, are not suitable to our aim. On the other hand, manual generators [2], like authoring tools, are not able to solve the problems they allow to create nor to propose any diagnosis of the learner's answers or help functionalities.

An AMBRE ILE is based on a knowledge based system which relies on a problem solver and allows to provide the learner with help, a diagnosis of his answers and explanations concerning his errors [3]. The problems proposed to the learner must be understandable by the solver to allow the ILE to provide these functionalities.

With AMBRE-teacher, teachers can influence the problems to be generated, by specifying a set of constraints on the exercises to generate. As the problems are built by the system from these constraints, the result of the generation will not only be a wording in natural language, but also a model of the problem usable by the solver.

3.1. The problem generation environment for the teacher

We designed and implemented a tool for problem generation dedicated to teachers, for the word additive problems domain suited to AMBRE-add used in primary school. Problems of this domain describe concrete situations, for example a marbles play: "Alex had 32 marbles. At the end of play, he has 45. How many marbles did he win?". This domain has been widely studied in mathematical didactics and several problems classifications have been established. The one used in AMBRE-add is presented in [3].

For word additive problems, constraints are of four types. The **structure** of a problem to be generated corresponds to its class, defined by several attributes that can be set or not. **Surface features** are the elements that complete the produced wording. The teacher can specify some elements of this category, for example themes, objects and characters. He can also choose the **values** of the data that will be used in the problems or define an interval for each required values and the wanted difference between min and max values, allowing the carrying over or not, etc. **Complication** concerns all options proposing to complicate the wording of the problem to adapt it to the students' level. Designing this part required a close collaboration with teachers to identify their needs. The environment proposes language complications and complications of the wording itself. For word additive problems, complication takes the form of vocabulary used and turn of phrases complexity, writing of numbers in full, modification of the sentences order, addition of distractor sentences, addition of non pertinent data. Not all constraints are mandatory for the exercises creation. Constraints not specified by the teacher will be randomly defined by the system.

The four categories defined for word additive problems are not reusable as it for another application domain of AMBRE. Nevertheless, structure features, surface features, and probably complication will still be necessary. Values will only be present for numerical domains.

3.2. The GenAMBRE architecture

The problem generation process that we established in the GENAMBRE architecture takes as input the set of constraints specified by the teacher and gives as output two elements: the wording of the problem in natural language for the learner and a computer-usable formulation of the wording named descriptive model of the problem, for AMBRE problems solver.

The problem generator architecture for AMBRE is presented in **Figure 1** and each of its components is presented in the following of the section. For a D domain (for example the additive word problems domain), the five knowledge bases of the *domain level* required by the *generation level* must be defined by using the domain independent formalisms of knowledge representation. The problem generation process is done in

two stages: the system builds a problem generation model, then builds the wording in natural language and the descriptive model of the problem. Both these processes are domain independent. Both processes and the knowledge bases of the D domain, constitute together a problem generator for the D domain: GenAMBRE-D.

Classification knowledge For each application domain, an expert gives to AMBRE solver, and consequently to GenAMBRE generator, a problems classification graph. This hierarchy of classes is used by the solver to classify the problem. This is a domain dependant class hierarchy, but its representation is the same for all domains.

Knowledge of the themes To generate a problem, it is necessary to know the concerned theme and the associated surface features (for example objects, characters and actions). Knowledge of the themes is given by the expert, or created by the teacher himself, through the surface features management module of AMBRE-teacher.

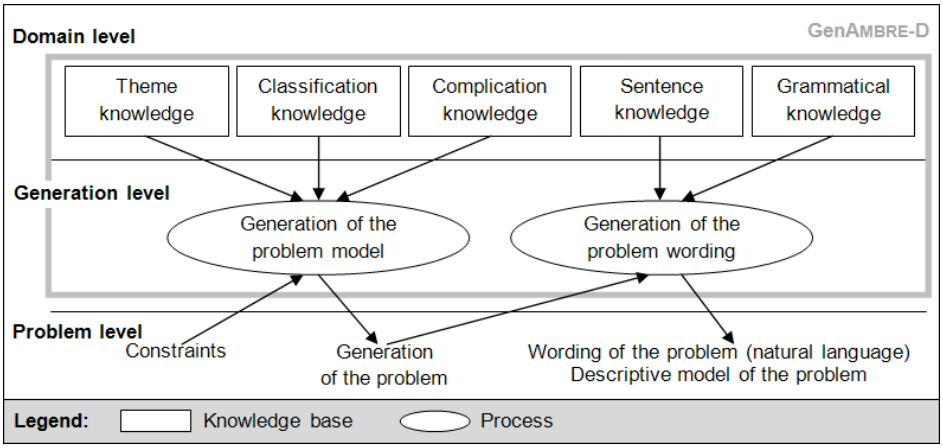

Figure 1. The GenAMBRE architecture.

Complication knowledge For additive problems, complicating a wording mainly consists in changing the sentences order and adding distractor sentences. So complication knowledge answers the following questions: how can one modify the order of the sentences of the problem? What distractor sentences can we add to problems and where can we place them?

Generation of the problem model process From the three knowledge bases previously described and from the constraints keyed in by the teacher, the system has to generate what we call a generation of the problem model. This model is an extensive descriptive model, because it also details the problem class and its theme. To build this model, the process fulfills the constraints defined by the teacher, notably by choosing random values for the undefined constraints.

Grammatical knowledge The domain expert must furnish a grammar to the generation system: a set of sentences structures that could be used in the domain.

Knowledge on the sentences The generated sentences, notably their structure, depend on the class of the problem. It is therefore necessary to know what sentences structures (from the grammar) could be used for the problem to be generated to permit

to generate the wording in natural language. So, knowledge on the sentences allows to associate the class of the problem to the usable sentences structures, and the associated elements of the problem.

Generation of the problem wording process To generate a wording in natural language, the process uses knowledge on the sentences and domain grammar, as well as the generation of the problem model previously created. Knowledge on the sentences allows the system to take conceptual decisions (deciding what to tell), then the process goes to the text generation step and establishes syntactical treatments (deciding how to tell it) and lexical and morphological ones (deciding how to write it).

4. Conclusion

In this paper, we presented how we designed a module dedicated to teachers for the AMBRE-add ILE, to allow them to adapt the ILE to the learning context and to their pedagogical approach. By adopting a generic approach, we identified, with the help from teachers, functionalities that a teacher module must propose for an AMBRE ILE (AMBRE-teacher). We have also enabled the teacher to configure the environment, generating problems suited to his needs, creating learning sequences suited to his students by choosing the problems and the behavior of the ILE, assigning these sequences to his students, and creating new themes of exercises.

These functionalities are implemented for the word additive problems domain. For this, we designed a problem generation system whose architecture is domain independent. Even if we integrated teachers in the design of AMBRE-teacher, it is now necessary to evaluate it in actual classroom situations with a significant number of teachers. Finally, we must also validate the functionalities defined for AMBRE-teacher and the genericity of the GenAMBRE architecture by implementing a teacher module for an AMBRE ILE for another domain.

5. References

[1] Burton, R. R. (1982). Diagnosing bugs in a simple procedural skill. Intelligent Tutoring Systems. London, Academic Press, 157-184.
[2] David, J.-P., Cogne, A. and Dutel, A. (1996). Hypermedia exercises prototyping and modelising. Computer Aided Learning and Instruction in Science and Engineering. S. B. Heidelberg, 252-260.
[3] Duclosson, N. (2004). Représentation des connaissances dans l'EIAH AMBRE-add, TICE'2004 conference, France, 164-171.
[4] Giroire, H. (1989). Un système à base de connaissances pour la génération d'exercices dans des domaines liés au monde réel, PhD Thesis, Université Paris 6.
[5] Jean-Daubias, S. (2009). Differentiated design: a design method for ILE. Research report RR-LIRIS-2009-015.
[6] Mitrovic, A., Stoinemov, L., Djordjevic-Kajan, S. (1996). INSTRUCT: Modelling Students by asking questions, User Modelling and User-Adapted Interaction, vol.6-4, 273-301.
[7] Nogry, S., Jean-Daubias, S., Duclosson, N. (2004). ITS Evaluation in Classroom: The Case of AMBRE-AWP, ITS'2004 conference, 511-520.
[8] Pecego, G. (1998). SYGEP, un Système de Génération d'Énoncés de Problèmes dans des domaines variés, PhD Thesis, Université Paris 6.
[9] Vivet, M. (1990). Uses of ITS: Which role for the teacher?, New Directions for Intelligent Tutoring Systems, NATO ASI series, Vol. F91, Springer-verlag, Sintra.

Building Resources for an Open Task on Question Generation

Vasile RUS[a], Eric WOOLLEY[a], Mihai LINTEAN[a], and Arthur C. GRAESSER[b]
[a]*Department of Computer Science*
[b]*Department of Psychology*
The University of Memphis
Memphis, TN 38152

Abstract. This paper contributes to the recent efforts to offer shared tasks on Question Generation by creating a data set of Question-Answer pairs collected from a community-based Question Answering repository. The data set was created with two shared tasks in mind: (1) question type generation and (2) open task. We have focused on the following six question types, which is a mix of shallow and deep questions: *how, who, what, when, where, why*.

Keywords. Question Generation, Data set, Open task

Introduction

The recent Workshop on The Question Generation Shared Task and Evaluation Challenge (www.questiongeneration.org) has identified four major categories of shared tasks that could help boost research on Question Generation (QG) (see *Chapter 2* in [5]). One category is *Text-to-Question* which covers shared tasks in which the input is raw or annotated text and the goal is to generate questions for which the text contains, implies, or needs answers. Every reasonable or good question should be generated. It is beyond the scope of this paper to discuss what a good question is. A special case of this category is a task in which any, not necessarily every, good question is generated. We will call such a task an *open task*. That is, the generation of questions is not restricted in any way, it is open. An example of a more restrictive task would be to generate questions of a certain type, e.g. *how* questions. In this paper, we describe our efforts to build a data set that could be used in an open QG task as well as in a more restrictive task. The data set is in the form of question-answer pairs (Q-A pairs) collected from Yahoo!Answers, the community-based Question Answering service that Yahoo offers to its users ([3]). In Yahoo!Answers, users can freely ask questions which are then answered by other users. The answers are rated over time by various users and the expectation is that the best answer will be ranked highest after a while.

The task of selecting the *question type* implies only determining the type of question that best suits the given input (raw or annotated) without really generating the corresponding question. For instance, a fragment of text which describes a procedure, i.e. a set of steps, would most likely trigger a *how* question. Question type selection is a *subtask* of the full QG task because it is a step in the overall QG process [2, 5]. The overall QG process is usually regarded as a 4-step process: (1) deciding when to ask a

question, (2) content selection, (3) question type selection, and (4) question construction. In our work presented in this paper we collected Q-A pairs corresponding to the following six types of questions: *how, who, what, when, where,* and *why*. Thus, the data set could be used in a subtask of question type selection that focuses on these six types of questions.

Data Collection

The process of collecting data for an open task on Question Generation involved the following steps: (1) identifying or creating a good source for efficient collection of texts and associated questions, (2) automatically collecting Q-A pairs, (3) automatically filtering the Q-A pairs, and (4) high-quality manual filtering.

1. Identifying or Creating a Data Source

Yahoo!Answers is a good source for data to be used in QG shared tasks for two reasons: (1) it contains open domain/general knowledge content and (2) both questions and associated snippets of text are available. In a recent survey among QG researchers, the first author of the paper found that an open/general knowledge source of data is preferred, namely texts from Wikipedia, as opposed to other types of sources. One problem with Wikipedia is that only texts are available and no associated questions. Creating questions for Wikipedia texts would be an extremely time consuming and expensive exercise. Yahoo!Answers is a good alternative to Wikipedia as a source of open domain/general knowledge QG data. The advantage of Yahoo!Answers is the availability of questions for given text fragments which means that the expensive step of generating the questions is avoided. On the other hand, only one question is available for each answer. Ideally, multiple good questions of various types should be available for a text fragment/best answer. It should be noted that Microsoft offers a similar service to Yahoo!Answers. The only reason we started with Yahoo!Answers is our familiarity with it and the availability of a public programming interface.

2. Automatic Collection

The second step in creating the dataset was to collect a large number of questions and answers. We decided to collect Q-A pairs for the following six types of questions: *how, what, when, where, who,* and *why*. These types include factoid or shallow questions (*who, when*) as well as deep questions (*how, why*). Ideally, we would obtain a balanced data set in which the same number of instances will be collected for each of the six types of questions. The balanced data is best for evaluating and comparing different approaches to a given task, e.g. QG in our case. Our balanced data set could be eventually extended to reflect a real distribution of question types and thus serve other evaluation purposes. Another important aspect of our data collection efforts was to identify questions on topics and disciplines of general interest so that the data set is attractive to many research groups.

Following the above guidelines, we started collecting Q-A pairs from Yahoo!Answers. In Yahoo!Answers, questions are characterized by the following three parts: question summary (usually a single sentence), question content (a paragraph that details the question and its context), and the chosen answer, which can be of variable length. The length of the chosen answer varies from one single sentence to one full page of text. Questions in Yahoo!Answers are categorized based on their topic. Examples of categories of questions are *Allergies*, *Dogs*, *Garden & Landscape*, *Software, and Zoology*. In order to collect an initial set of Q-A pairs, a number of 244 Yahoo categories were queried on all six types of questions. To collect questions of each type we used the *wh*-words that define the question types (e.g. *where*, *what*, *how*, etc.) and searched for questions in Yahoo!Answers that contain those keywords in their summary. Table 1 lists some example of question summaries that were extracted from Yahoo!Answers. A maximum of 150 questions was downloaded per each category and question type, resulting in a total of maximum 150*244*6 = 219.600 number of candidate questions to be collected. Because not all categories had at least 150 answered questions of certain type, the number of collected questions is actually smaller.

Table 1. Examples of questions, one from each of the six categories.

Category	**Type**	**Question Summary**
Add-ons	*How*	*How important is to have a mouse pad?*
Aircraft	*How*	*How do pilots of small aircraft know how far they are from an aerodrome?*
Economics	*Why*	*Why did the social and economic status change during the Middle Ages?*
Law & Ethics	*Who*	*Who wrote the final copy of the Stabilization Act of 2008?*
Radio	*Where*	*Where do radio stations get their digital music from?*

3. Automatic Filtering

Upon review of the collected Q-A pairs in the previous step, it was clear that some of the data would not be appropriate to keep. The following criteria were applied to filter out Q-A pairs. The *first* criterion to be established was question length. In this case, length is described as the number of words in a given question. Questions in the Q-A pairs are denoted by the *Subject* tag. A minimum requirement of 3 words was proposed. This represents the smallest "common" length of a valid question. This corresponds to a typical definition questions such as *What is (object of choice)?* That is not to say that valid questions with less than 3 words do not exist. However, for the purposes of this dataset 3 words was set as the minimum. The same reasoning was applied to the answer data in the Q-A pairs. The answer data should be of a required minimum length. In our case, that length was set to 10 words. Again, valid answers can, of course, contain fewer words. However, for the purposes of this dataset and its intended uses, a minimum length of 10 words was selected.

A close review of the data also showed that it had content that was somewhat less than what might be perceived polite and courteous. Specifically, curse words, words that are and/or refer to sexual explicitness, and content that contained ethnically intolerant words. Therefore, a *second* criterion to filter out Q-A pairs was developed such that Q-A pairs with bad content were discarded. A "bad word" list was generated from the information gathered at two websites [6, 7]. When combined, the list totals

over 850 words. Unfortunately, this list is not complete, and it would be a considerable challenge to maintain a current listing of such terms given the dynamic nature of language. As such, a final decision with respect to including a Q-A pair into the final set must be made manually, i.e. by a human (see next subsection). It should also be noted that the "bad word" list does not include any misspellings of the terms therein, as it would be very difficult to anticipate all of the incorrect ways to spell these terms.

Other criteria were evaluated and rejected, e.g. Q-A pairs that were answered by the highly rated Yahoo!Answers users or the total number of answers per given question is another method of filtering that was considered.

The total reduction due to filtering in the dataset was about 55%.

4. High-Quality Manual Filtering

In the fourth and final step, we used three human raters to filter out the Q-A pairs resulted from the previous automatic filtering phase. The raters have worked on different subsets of the collected and automatically filtered Q-A pairs. This is because the goal was to collect as many Q-A pairs as possible. We will use different raters to judge same Q-A pairs once we have large enough initial data set. Inter-rater agreement scores will be reported.

In order to facilitate the human filtering step, we built a software tool that allows a quick analysis of each question and the corresponding content and answer by the human rater. Additionally, the tool allows easy relabeling and removal of the corresponding Q-A pair in case it is deemed unacceptable (incorrect, improper, or too difficult for the purposes of the data set which are question type detection and generating the best question for a given answer).

Even with the aid of the tool, manual filtering proved to be a time-consuming process. On average, it takes about 10 hours to select about 100 questions. The advantage is that it results in high quality Q-A pairs for the two QG tasks for which we developed the data set. Similar to automatic filtering, manual filtering further reduces the collected data set. On average, out of all the Q-A pairs judged by humans about 10% of them are retained. For some types of questions and topics, such as *when* questions and *Camcorders*, the retaining rate is even lower than 10% at 2%. This is because the majority of *when* questions in the Camcorders category are not really *when* questions. The keyword *when* is used frequently to describe a context while the actual main question is usually of a different type as illustrated by the following example: *When I transfer footage from my video camera to my computer why can't I get sound?* In this example, *when* introduces the context of the main *why* question. The only category with a high retaining rate for *when* types of questions is the *History* category. A similar problem of low retaining rate during human filtering was noticed for *where* questions. *Where* can be used to ask for a source of information, e.g. a website or book, and not necessarily for a real place. An example of a *where* question which asks for an information source is the following: *Where can I find information about the Industrial Revolution in USA?* While such *where* questions could be retained we opted not to.

We list below examples of reasons for which Q-A pairs were discarded during manual filtering.

1. **The question is a compound question.** For instance, the following question contains both a *how* and an *who* question: *How do you figure out who your video card manufacturer is?* Another example is the question *How long has*

the computer mouse been around, and who is credited with its invention ?

2. **The question is not in interrogative form.** An example of a non-interrogative question is the following *I want a webcam and headset etc to chat to my friends who moved away?*
3. **Poor grammar or spelling.** The following questions are examples of questions with poor grammar and spelling: *Yo peeps who kno about comps take a look?* and *Who the memory eaten is bigger?*
4. **The question does not solicit a reasonable answer for our purposes.** An example of such a question is *Who knows something about digital cameras?*
5. **The question is ill-posed**. For instance, for the question *When did the ancient city of Mesopotamia flourish?* the answer is *Mesopotamia wasn't a city.*

Future Work and Conclusions

The initial data set we collected so far contains about 500 Question-Answer clean pairs (automatically and then manually filtered) selected out of a set of almost 5000 candidate Q-A pairs. The goal is to increase the size of the clean dataset to 5000. We already have enough raw Q-A pairs collected and automatically filtered. The only challenge to increase the size of the clean dataset is to manually filter more Q-A pairs. Once we have achieved our goal of 5000 clean Q-A pairs we plan to further validate the Q-A pairs and to annotate the answers with a deep representation. For further validation, we envision an experiment in which human subjects are shown first the answer and then the corresponding question and asked to rate how good they consider the question given the answer. We are considering three options for annotating the answers with a deep representation: FrameNet [1], PropBank [4], or Unified Annotation Language [9]. In a recent survey among QG researchers, PropBank seems to be the preferred annotation language.

One important conclusion to draw from our work is that collecting data from community Question Answering sources is more challenging than it seems. The alternative of explicitly collecting Q-A pairs from sources of general interest such as Wikipedia through target experiments with human subjects may be a comparable rather than a much costlier effort.

References

[1] C. Fillmore, C.F. Baker & H. Sato. The FrameNet Database and Software Tools. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC)*. Las Palmas. 1157-1160, 2002.

[2] R. Nielsen. (2008). Question Generation: Proposed challenge tasks and their evaluation. *Proceedings of the Workshop on the Question Generation Shared Task and Evaluation Challenge.* NSF, Arlington, VA.

[3] T. Marciniak, Language generation in the context of Yahoo! answers. *Workshop on the Question Generation Shared Task and Evaluation Challenge.* NSF, Arlington, VA, 2008.

[4] M. Palmer, P. Kingsbury, D. Gildea. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics* **31** (1): 71–106, 2005.

[5] V. Rus & A.C. Graesser, *Workshop Report: The Question Generation Task and Evaluation Challenge*, Institute for Intelligent Systems, Memphis, TN, ISBN: 978-0-615-27428-7.

[6] http://www.noswearing.com/list.php

[7] http://carbonize.co.uk/Old/sexglos.php

[8] http://www.cs.brandeis.edu/~jamesp/ula2007/index.html

Influence of Reading Goals on Question Generation

Vicente SANJOSÉ[1], Koto ISHIWA[2], José OTERO[2]
[1] *Polibienestar, Universidad de Valencia (Spain). C/ Alcalde Reig 8, 46006-Valencia-Spain, vicente.sanjose@uv.es*
[2] *Departamento de Física, Universidad de Alcalá, 28871 Alcalá de Henares, Madrid, Spain, jose.otero@uah.es, ishiwakoto@hotmail.com*

Abstract. This study investigates the generation of information-seeking questions when subjects process scientific texts with a particular goal in mind. Reading goals were manipulated through two different tasks: reading for understanding vs. reading in order to solve an algorithmic problem. In these situations, a reader's immediate goal is to create an internal representation of discourse that is appropriate for the required task. The questions asked correspond to obstacles found in building this representation. Three categories of questions were distinguished, associated to three types of obstacles: association questions, explanation questions, and prediction questions. The results show that obstacles in building mental representations, and the corresponding questions asked, depend on reading goals. First, significantly more explanation questions were asked in the understanding condition than in the problem solving condition. Second, goal condition was found to differently influence association questions and explanation questions: effect size was much larger for explanation questions than for association questions.

Keywords. information seeking questions, scientific texts, reading goals.

Introduction

Computational models of question generation (QG) do not necessarily have to mymic human questioners. However, a consideration of human question asking may probably help in the design of automated QG by identifying variables that influence the type and quality of questioning. For instance, take a task such as generating model questions on a school text. What are key variables in quality questioning by humans in this situation? Quality questions may depend on text, subject, and task variables that should also be considered by automatic systems that aim at quality questioning.

In this study, we address the generation of questions by humans who process science texts. We focus on questions addressed to solve knowledge deficits, i.e., "information seeking questions" (ISQs). These questions have been studied from many points of view in education and psychology. However, the process of *generating* ISQs has been less studied. Goals and obstacles appear as key elements in this process: the generation of ISQs by humans may be conceptualized as a request for information in order to remove obstacles towards a certain goal [1]. The immediate goal of a reader consists in creating an internal representation of discourse appropriate for the attempted task. Obstacles may be found in this attempt and ISQs may be asked to overcome them.

Building an internal representation consisting of a situation model involves generating inferences that elaborate the textbase. Therefore, the obstacles found when readers try to build a situation model should correspond to difficulties in generating these inferences. In a situation of conscious understanding, three broad classes of inferences that may occur have been identified: associative, explanatory, and predictive [2]. These inferences correspond to the three types of questions that may be asked when readers try to build a situation model: questions linked to associations, explanations, and predictions. The first kind of questions addresses the need to adequately represent the entities of the system under consideration, as well as their properties. These may correspond to *who*, *what*, *how*, *when* and *where* questions (i.e. '*How does the boom work on a sailing boat*'?). The second kind of questions focuses on justifications or explanations for these entities. It basically consists of *why* questions (i.e. '*Why can a boat sail against the wind*?'). Lastly, questions addressing the need to foresee consequences correspond to *what happens next* or to *if-then* questions (i.e. '*What will happen if the wind is really strong?*').

Readers' external tasks, i.e., their mediate goals, may influence the immediate goal of creating a discourse representation with certain characteristics. Therefore, reading goals should also influence the obstacles found in creating these representations and the questions asked. We manipulated representation goals through two different tasks: reading for understanding vs. reading in order to solve a problem. Causal relations are important in mental representations of expository texts that are read for understanding [3]. However, they are expected not to be so important when one builds a representation in order to solve an algorithmic problem. Therefore, more explanation questions (T2) would be expected when one reads for understanding than when one reads in order to solve an algorithmic problem. We did not have clear predictions regarding association questions (T1) in both situations. As predictive inferences are rarely made when reading these texts [4], we expected few prediction questions (T3) in any of the two conditions.

The study included three similar experiments that provided consistent results [5]. In this article, we report one of these experiments only.

1. Method

1.1. Subjects

Sixty eight students majoring in Biology at the University of Alcalá (Spain) participated in the study. The experiment was included as a part of the activities in an introductory physics course, and 0.3 points were added to the students' grades depending on performance in the experiment.

1.2. Materials

A booklet containing two passages was provided to the students (one of these passages is shown in Table 1). The first two sentences introduced a certain physical setting. The third sentence, starting with "However…", presented a phenomenon related to this setting, at variance with common sense beliefs. The fourth and fifth sentences elaborated on this phenomenon and provided data needed to find a solution in the problem solving condition. The passages were the same in both experimental

conditions, except for the inclusion of a sentence requesting a calculation in the problem solving condition.

Table 1. Example of the passages used in the experiments in both experimental conditions

	Sailing
Instructions	*Read the description of the following phenomenon carefully, trying to understand it.*
	Sail boats are used since ancient times. Wind collides against the surface of sails that push the boat so that it navigates. However, sail boats are able to navigate against the wind since several centuries ago. When they navigate against wind sail boats are able to reach speeds up to two times the square root of the wind's speed at that moment. An appropriate wind to navigate may have a speed of 50 km/h.
Understanding version:	*State any query or question that you may have so that you may correctly answer the questions of the comprehension test next day.*
Problem solving version:	Calculate the speed of the boat in this case. *State any query or question that you may have so that you may correctly solve the problem, next day.*

1.3. Procedure

In the understanding condition the subjects were informed that they should read the two passages in order to understand them in preparation for a test on the passages' content. The test would be administered in the following session. In the problem solving condition, the subjects were informed that they had to read the two passages in order to solve the problems in the following session.

In any of the two conditions, students had instructions to read the passages at their own pace, and to ask questions in writing in the space provided. They were informed that an answer to these questions would be provided in writing prior to the testing session. These answers could be used if needed in the test. The complete procedure took about 40 minutes. At the end, the subjects were debriefed informing them that there would not be a second session.

1.4. Measurements

The questions asked were classified as association questions (Type 1), explanation questions (Type 2), and prediction questions (Type 3). The Kappa coefficient of intercoder agreement was .92

As the assumptions on normality and homogeneity of variance for the number of questions asked were found not to be held a Mann –Whitney U test was used to make comparisons across goal conditions.

2. Results

Table 2 shows the average number of questions per subject and per passage, standard deviations (in parenthesis), and effect sizes.

Table 2. Questions and explanatory statements per subject and per passage, standard deviations (in parenthesis), and effect size. (†: p< 0.01).

	Understanding	Problem solving		Effect Size
T1	.76 (1.10)	.68 (.81)		.08
T2	.80 (.65)	.18 (.38)	†	1.16
T3	.09 (.26)	08 (.18)		.04

There were more questions of all kinds in the understanding condition than in the problem solving condition, but the difference was significant for explanation questions only.

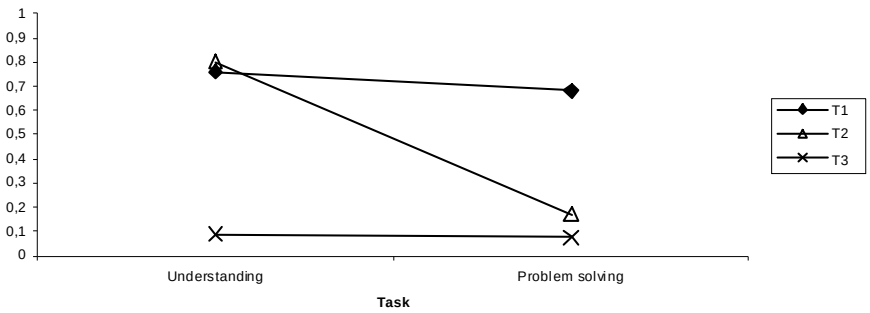

Figure 1. Average number of questions per subject and per passage in the understanding condition and the problem solving condition.

3. Discussion and conclusions

Consistently with our hypotheses we found that obstacles in building mental representations, and the corresponding questions asked, depend on subjects' goals. First, there was a positive difference in asociation questions between the understanding condition and the problem solving condition, although it did not reach statistical significance in this experiment. The fact that differences in the same direction were found in other experiments [5] points toward the practical significance of the result.

Second, there was a significant effect of the reading goal on the number of explanation questions. Subjects make significantly more explanation questions in the understanding condition than in the problem solving condition, with a remarkable effect size.

Third, goal condition was found to differently influence association questions and explanation questions, as shown by effect sizes. Representing entities seems to be a more stable requirement than explaining them. This is in agreement with a proposal of a canonical ordering of question asking during knowledge exploration: questions on definitional information about entities precede causal, explanation questions [6].

Finally, prediction questions were too few to show potential differences between conditions, as expected from previous research.

In sum, the study provides evidence about the dependence of readers' questioning on processing goals. Therefore, artificial systems that pay some attention to human questioning should include reading purpose as a significant parameter in question generation on texts. Generated questions do not depend on textual input and the system's knowledge base only. In addition to these variables, they directly depend on the internal representation that the system tries to build and, indirectly, on the system's reading goal.

References

[1] Otero, J. (in press). Question generation and anomaly detection in texts. En D. Hacker, J. Dunlosky, & A. Graesser (Eds.), *Handbook of Metacognition in Education*. Routledge. In press.

[2] Trabasso, T., & Magliano, J. P. (1996). Conscious understanding during comprehension. *Discourse Processes, 21*, 255-287.

[3] Coté, N., Goldman, S., & Saul, E. U. (1998). Students making sense of informational text: Relations between processing and representation. *Discourse Processes, 25*, 1-53.

[4] Millis, K., & Graesser, A. (1994). The time-course of constructing knowledge-based inferences for scientific texts. *Journal of Memory and Language, 33*, 583-599.

[5] Ishiwa, K., Sanjose, V., & Otero, J. (2009). *Generation of information-seeking questions on scientific texts under different reading goals*. Manuscript submitted for publication.

[6] Graesser, A. C., Langston, M. C., & Bagget, W. B. (1993). Exploring information about concepts by asking questions. In G. V. Nakamura, R. M. Taraban, & D. Medin (Eds.), *The psychology of learning and motivation: vol 29, Categorization by humans and machines* (pp. 411-436). Orlando, Fl: Academic Press.

Increasing Problem Simplification as Scaffolding in Exercises

Tsukasa HIRASHIMA
Hiroshima University

Abstract. In this paper, three types of increasing problem simplification, (1) formulation partialized problem, (2) solution partialized problem and (3) specialized problem are introduced. They are defined as problems that can be solved as sub-process of the original problem. In this paper, a model of problem solving process is proposed. Based on the model, then, the three types of increasing problem simplification are described. As an evaluation, the problems generated by the method are examined by comparing the problems generated by human tutors and practice books. Several ways to help students with the simplified problems are also explained.

Keywords. Increasing Simplification, Problem Generation, Problem Simplification, Scaffolding

Introduction

Problem solving practice plays a crucial role in enhancing the students' problem-solving capabilities. Developing an advanced and elaborate form of problem practice is, therefore, one of the most important issues in the research of the computer-based learning environments. In problem solving exercises, a student often fail to solve a problem. In that case, teaching the correct solution is not always an effective way to help the student because he/she may passively accept the solution without trying to check it or regenerate it [1]. Therefore, how to help a student solving a problem him/herself is an important issue. Polya suggested that using problems which are generated by simplifying the original difficult problem is one of the most promising methods to realize such help [2]. This method is actually popular in one-on-one human tutoring. Purpose of this research is to realize this method in computer-based learning environments.

The simplest way to realize this support is to prepare various simplified problems to each original problem used in the problem exercise beforehand. However, it is not easy task to prepare enough number of problems. Moreover, when the student fails to solve even the simplified problems, simpler problems are required. Based on these considerations, it is hard to prepare every necessary simplified problem beforehand.

Automatic problem generation from the original problem is a promising approach to realize effective use of the simplified problems. In this research, first, a model of problem-solving process that describes the process as a series to change problem structure is proposed. Simplified problems are defined as a problem which can be solved by (i) partialized process and (ii) specialized process of the original problem only.

Section 1 describes a framework for characterizing problems based on sub-processes in and intermediate structures created during the problem-solving process. By using the description, three types of increasingly simplified problems are defined. Section 2 evaluated the framework through two experiments, one is problem simplification by human tutors, and the other is examination of problems in several practice books. In Section 3, the ways to help a student in problem-solving with increasingly simplified problems are proposed. Currently, only mechanics problems in high school are used as examples. To confirm the applicability of this research to other domains is a future work.

1. Definition of Increasingly Simplified Problems

This section describes a framework to characterize a problem with the following three components; (1) surface structure, (2) solution structure, and (3) constraint structure [3,4]. Next, three types of simplified problems are described; (I) formulation partialized problem, (II) solution partialized problem and (III) solution specialize problem.

1.1. A Model of Problem-Solving

Several investigations have indicated that the formulation of a problem is often the most difficult task for students in problem-solving. This suggests that the formulation process is important to manage the difficulties of problems. Several models of problem-solving represent the formulation process as refining surface feature s of the problem [5,6]. In this research, the surface features are described with a semantic network structure of objects, their attributes and relations between the objects. This network is called "surface structure". Mover, refinements are regarded as changes in the surface structure to which numerical relations can be applied. For example, in mechanics, when a problem includes statements such as "smooth incline", there is an object "incline" in the surface structure with attribute "smooth". To apply numerical relations to the surface, the "smooth" should be changed to "frictional coefficient zero". The structure to which numerical relations can be applied is called "formulated structure" and the process to make the formulated structure "formulation process". Numerical relations are applied on the formulated structure and the required value is derived. This process is called "calculation process" and a structure including the required value is called "goal structure".

Based on this view, the problem-solving process can be divided into three phases; (1) statement-understanding process, (2) formulation process, and (3) calculation process, as shown in Figure 1. Because the task of the formulation process is to reduce and remove differences between the surface structure and the formulated structure, the surface structure is useful to characterize the formulation process. The task of the calculation process is to derive the required value by applying a series of numerical relations. The series of numerical relations is called "solution structure" and use it to characterize the calculation process in this paper.

Numerical relations which don't contribute to solve a problem but which exist in the situation set up by the problem are also important to characterize the problem. For example, numerical relations according to kinetic energy aren't necessary to solve Problem-1 shown in Figure 2, but the numerical relations are necessary to solve Problem-2 posed in the same situation. Therefore, numerical relations included in the

situation are also important to characterize problems. The situation is characterized by a network composed of numerical relations among attributes included in the situation. The network is called "constraint structure" in this research.

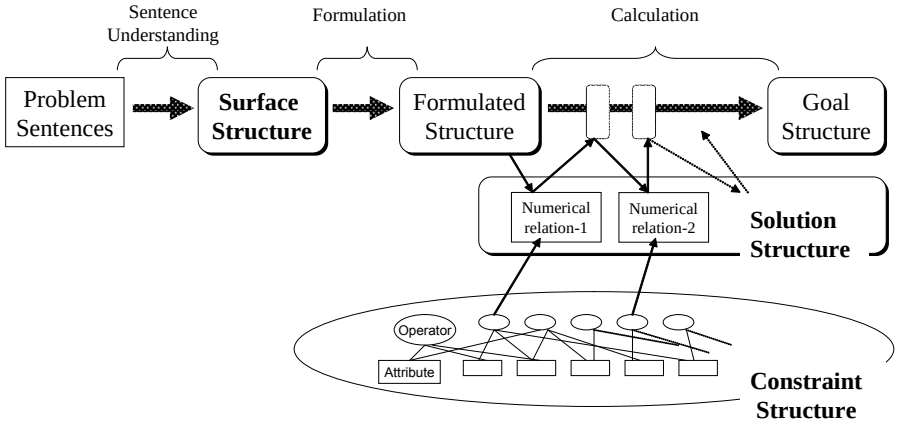

Figure 1. A Model of Problem-Solving Process.

[Problem-1] A block of mass *M* is put on a smooth incline quietly. The angle of the incline is *q* and the gravity acceleration value is *G*. Find the force of the block in parallel direction to the incline.
[Problem-2] A block of mass *M* is put on a smooth incline quietly. The angle of the incline is *q* and the gravity acceleration value is *G*. Find the kinetic energy of the block after *T* seconds.
[Problem-3] A person, who is going up in an elevator moving with velocity *V0*, releases a ball. Find the velocity of the ball after *T* seconds. The gravity acceleration value is *G*.
[Problem-4] The ball is thrown with initial velocity *V0* to the upper vertical direction. Find the velocity of the ball after *T* seconds. The gravity acceleration value is *G*.
[Problem-5] A block of mass *M* is put on a smooth incline quietly. The angle of the incline is *q* and the gravity acceleration value is *G*.
(5a) Find the force of the block in parallel direction to the incline.
(5b) Find the velocity of the block when it moved for a distance of S on the incline.
[Problem-6] A block of mass *M* is put on a coarse incline. The angle of the incline is *q* and the gravity acceleration value is *G*. The coefficient of friction between the block and the incline is m. Find the acceleration of the block in parallel direction to the incline when its initial velocity is *zero*.
[Problem-7] A block of mass *M* is put on a coarse incline. The angle of the incline is q and the gravity acceleration value is *G*. The coefficient of friction between the block and the incline is m. Find the acceleration of the block in parallel direction to the incline when its initial velocity is *V* to the upper parallel direction on the incline.
[Problem-8] A block of mass *M* is put on a smooth incline quietly. The angle of the incline is *q* and the gravity acceleration value is *G*. Find the longest moved distance to the upper parallel direction on the incline when the initial velocity of the block is *V* to the upper parallel direction on the incline.

Figure 2. Examples of Mechanical Problems.

1.2. Categorization of Simplified Problems

In this paper, a problem is characterized by a surface structure, solution structure and constraint structure. Then, simplification of an original problem is carried out by reducing the problem-solving process or by specializing in the numerical relations used in the process. The reduction of problem-solving process is carried out by formulating the surface structure or partitioning of the solution structure. Formalizing of the surface structure means reducing of the formulation process. A problem that can be generated by formalizing the surface structure of the original problem is called "formulation partialized problem". Partitioning of the solution structure means reducing of the calculation process. A problem that can be generated by partitioning the solution structure of the original problem is called "solution partialized problem". Specializing of the problem-solving process is carried out by specializing in the constraint structure. A problem that can be generated by specializing in the constraint structure of the original problem is called "solution specialized problem".

Problem-4 is a formulation partialized problem for Problem-3, that are shown in Figure 2. The surface structure of Problem-3 is Structure-A shown in Figure 3. To solve Problem-3, Structure-A should be changed to Structure-B where "elevator" is omitted and "velocity of the elevator" is transformed to "initial velocity of the ball". Structure-B corresponds to the surface structure of Problem-4. Because Problem-4 has the more refined surface structure than Problem-3, Problem-4 is easier than Problem-3 in the formulation process.

Problem-5a is a solution partialized problem of Problem-5b. Figure 4 shows the solution structure of Problem-5b. Derived denotes the attribute value is derived in the calculation process but isn't an answer of the problem. By changing several derived attributes to given attributes or required attribute, a part of the solution structure of the original one can be generated. The problem with such a partialized solution structure is simpler than the original problem in the calculation process. In the solution structure of Problem-5b, by changing "acceleration of the block in parallel direction" to the required attribute, a partialized solution structure is generated. Therefore, Problem-5a characterized by the partialized solution structure is a solution partialized problem of Problem-5b.

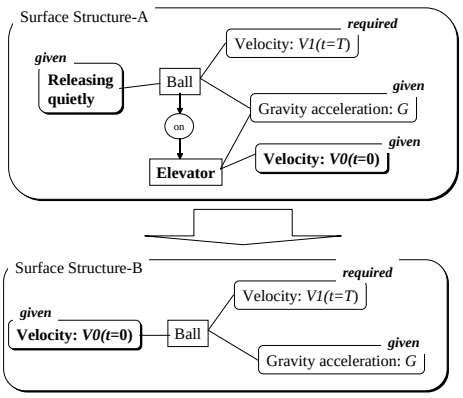

Figure 3. Simplification of Surface Structure.

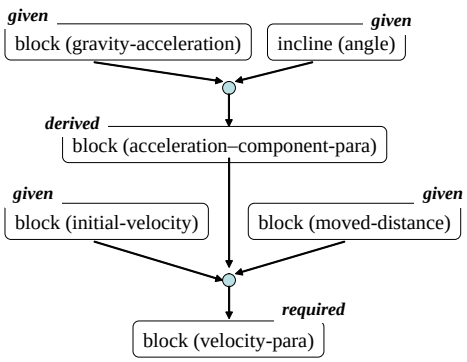

Figure 4. Solution Structure of Problem-5b

Problem-5a is a solution specialized problem of Problem-6. Figure 5 shows a part of the constraint structure of Problem-6. Simplification of the constraint structure is achieved by specifying an attribute value that can be omit it in the constraint structure. For example, when the value of the frictional coefficient becomes "zero", the frictional coefficient can be omitted in the constraint structure. Consequently, several numerical relations can be also simplified or omitted in the constraint structure. In Figure 5, when the value of the frictional coefficient becomes "zero" in the constraint structure, shown in Figure 5, the equation of motion is simplified from "ma = mg sin ϕ – μ mg cos ϕ" to "ma = mg sin ϕ". Such specialized numerical relations can be used only in the specialized situation where the frictional coefficient is zero. Then, the original numerical relations can also be used in the specialized situation. Therefore, Problem-5a characterized by the specialized constraint structure is a solution specialized problem of Problem-6.

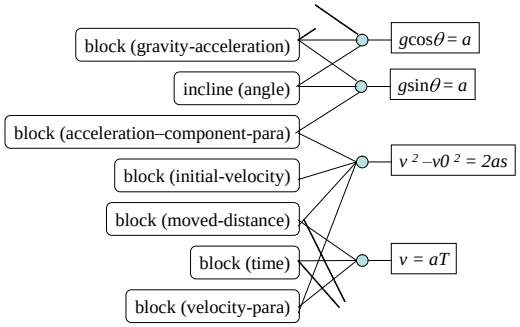

Figure 5. A part of a Constraint Structure.

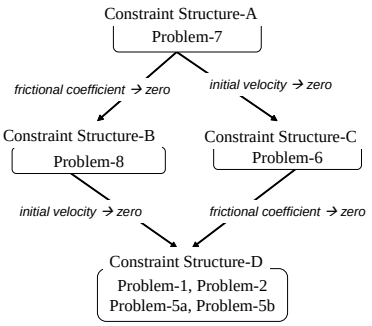

Figure 6. A Network of Constraint Structure.

1.3. Evaluation of the Categorization

As the evaluation of the categorization of problem simplification, problem pairs of the original problem and a problem that is supposed to help a student to solve the original one were gathered, and then, relations of the two problems in the pairs were examined corresponding to the definition explained in Section 1.2. The problem pairs were collected in two ways as follows: (1) problem simplification by human tutors, and (2) analysis of practice books. The pairs were examined with the following conditions;

When the two problems have the same solution and constraint structure, one of the two might be a formulation partialized problem. When the surface structure of one problem can be generated by formalizing the surface structure of the other problem, the former one is a formulation partialized problem for the latter one. Currently, four types of formulation operations are used in the examination, as follows: (a) changing a qualitative value of a quantitative value (for example, changing "putting quietly" to "initial velocity zero"), (b) adding a necessary attribute (indication of the necessity of tension "T" in the problem solving process), (c) rewording a give attribute to a necessary attribute (rewording "work load X" to "charge of energy X" in order to use numerical relation of energy), (d) shifting an attribute of one object to the attribute of the other object (shifting "the velocity of the elevator" to "the initial velocity of the ball" in Problem-3). When the two problems have the same surface and constraint structures, one of the two might be a partial problem. When the solution structure of

one problem can be generated by changing the derived attributes to require or given ones, the former problem is a solution partialized problem for the latter one. When the surf ace, solution and constraint structures of a problem can be generated by specializing in the three structure s of the other problem, the former one is a solution specialized problem for the latter one. Currently, the specialization means changing a value of an attribute to a value that makes it possible to omit the attribute in the constraint structure.

In the experiment of problem simplification by human tutor, fifteen college students with experience in tutoring were subjects. Assuming that their pupils couldn't solve a problem, the subjects were asked to simplify the problem to help the pupils. Three mechanical problems were provided for them and they generated 65 problems in total. All types of simplified problems were generated and only two problems were outside of the definition of the problem simplification in this paper.

In the analysis of practice books, the following three types of problem pairs were gathered: (1) a problem and a hint for it, (2) a problem and a reference problem for it, and (3) a pair of problem items (the pair of Problem-5a and 5b is an example). Four hundred eighty-seven pairs were gathered. Table 1 shows the results of the analysis. Seventy-four percent of the problems are categorized into the three types of simplified problems. Forty-seven referred problems shared a part of the solution structure with the original one. Eighty-four items had the same constraint structure but different set of given and required attributes for the original problems. These problems have also relation to the original problems but it is necessary to consider the context of the problem practice to judge whether or not they are useful to solve the original problems.

Through the analysis, we have confirmed that the definition of problem simplification is explained almost of the gathered problems and is also useful to analysis the problems outside of the definition. In Section 2, problem generation based on the definition is explained in more details.

Table 1. Analysis of Practice Books

	Hits(69)	Reference(120)	Item(308)
Formulation partialized	**30**	**18**	**0**
Solution partialized	**39**	**0**	**191**
Solution specialized	**0**	**52**	**29**
Solution partial shared	0	47	0
Given-required changed	0	3	84
Same required attribute	0	0	4

2. Generation of Simplified Problems

To realize the problem generation based on the definition of the problem simplification, it is necessary to prepare characteristic description for each problem. Although the quantity of description for each problem increases, because the description can be written independently of the context of problem practice, it is expected to reduce the load to prepare problems and their relations by hand.

2.1. Description of Problems

In order to realize problem generation, it is necessary to prepare (1) surface structure, (2) solution structure, (3) constraint structure, but also (4) problem sentences for the

surface structure, (5) formulated structure, (6) formulation operators which are used to explain the change from the surface structure to the formulated structure, and (7) problem sentences for the formulated structure. Currently, because modeling of formulation process is not enough, the description of the process of formulation process should be prepared.

Although the way to write constraint structure is clear, the number of numerical relations is often too many to write for each problem. However, one constraint structure includes many problems and there are not so many possible constraint structures in the domain of high school mechanics. So a network of the constraint structures can be prepared. Figure 6 show a part of the network of the constraint structures. In the situation of Constraint Structure-A, the block has the initial velocity and the frictional coefficient. Then, a specialized constraint structure by changing the initial velocity to zero is Constraint Structure-B. By using this network, the constraint structures do not have to be written for each problem. The research to generate the network itself with a method of automatic modeling is currently investigating.

2.2. Generation of Formulation Partialized Problem

For an original problem, prepared formulated problem is provided as the formulated partialized problem. The formulation operators are used to explain the change from the original problems to the formulated problem Therefore, a problem has only one formulation partialized problem. Because a formulated structure, however, usually related to several surface structures, when there are two surface structures related the same formulated structure and one surface structure is judged more formulated than the other, it is possible to use the former one as the formulation partialized problem for the latter one. Currently, the connection of the two problems should be written by problem author who prepare the problems.

2.3. Generation of Solution Partialized Problem

Solution partialized problems has the same surface structure as the original problem but the set of given attributes and required attribute is different. Therefore, a solution partialized problem is provided as an additional comment to the original problem. For example, the additional comment that "find first the acceleration of the block in parallel direction to the incline" to Problem-5b, changes the derived attribute "acceleration" to the required one, then original problem is changed to a solution partialized problem. If the value of the acceleration is given to Problem-5b, it is possible to omit to derive the acceleration in the calculation process. In this case, the derived attribute "acceleration" is changed to given attribute and the new problem is a solution partialized problem.

2.4. Generation of Solution Specialized Problem

In order to generate specialized problems, the network of constraint structure is used. For example, in Figure 6, Constraint Strucfture-1 can be specialized to Constraint Structure-2 by changing the initial velocity to zero, or Constraint Structure-3 by changing the frictional coefficient to zero. Because Problem-7 belongs to Constraint Structure-A, the solution specialized problems can be generated by specializing in the initial velocity to zero or the frictional coefficient to zero.

3. Support of Problem-Solving with Simplified Problems

In problem practice, when a student cannot solve a problem, it is useful to provide another problem that is simplified the original one. Usually, the student knows almost enough knowledge to solve the original one, but it is something insufficient to solve it. It is not always effective to explain the solution of the problem because most of the explanation is not necessary for the student. It is reasonable to assume that the student can solve a little simplified problem and the difference between the original one and the solvable problem is the cause of the failure of the problem-solving. Based on this consideration, providing simplified problems is a promising method not only to help to solve problems but also to promote learning from problem practice. In a basic way to use problem simplification, first, the formulation partialized problems are used. If a student can be solved the simplified problem, the student has to concentrate on the formulation or interpretation of the problem. If the student cannot solve the formulation partialized problem, there is something insufficient in the solution process. The solution partialized problems are, then, provided to the student. If the student can solve them completely, there is something insufficient in the combination of partial knowledge. When there is a solution partialized problem that the student cannot solve, a solution specialized problem should be provided. This means that the student don't have knowledge to deal with the original problem at this point in time. How to manage the simplified problem in practical situation is our important future work.

4. Concluding Remarks

In this paper, three types of increasing problem simplification, (1) formulation partialized problem, (2) solution partialized problem and (3) solution specialized problem, are defined. Because the simplified problems can be solved as sub-process of the original problem, it is possible to judge them as simplified ones without context of their learning. This research has been mainly investigated from the viewpoint of support facility of problem solving exercises. The discussion from the viewpoint of question generation is the most important remaining future work as well as the practical implementation and evaluation [7].

References

[1] VanLehn, K., R.M.Jones & M.T.H.Chi: A Model of the self-Explanation Effect, *Journal of the Learning Science*, 2(1), pp.1-59(1992)..
[2] Polya, G.: *How to Solve It*, Princeton University Press(1945).
[3] T.Hirashima, T.Niitsu, A.Kashihara, J.Toyoda: An Indexing Framework for Adaptive Setting of Problem in ITS, *Proc. of AIED'93*, pp.90-97(1993).
[4] T.Hirashima, A.Takeuchi: A Metadata Editor of Exercise Problems for Adaptive Problem Sequencing, , *Proc. of AIED2003*, pp.425-427(2003).
[5] Nathan, M.J. Kintsch, W., Young: A Theory of Algebra-Word-Problem Comprehension and Its Implications for the Design of Learning Environments, *Cognition and Instruction*, 9(4), pp.329-389(1992).
[6] Poltzner, p.: How Quantative Problem Solving in Mechanics Improves by Qualitative Reasoning, *Proc. of AIED'93*, pp.282-289(1993).
[7] Rus, V. & Graesser, A.C. (Eds.). The Question Generation Shared Task and Evaluation Challenge. (2009).

Generating Questions from OpenLearn study units

Brendan WYSE [a,1] and Paul PIWEK [a]
[a] *Computing Department, Open University, UK*

Abstract. OpenLearn is a large online educational repository which can be used as a resource for developing and testing Question Generation (QG) systems. OpenLearn is described in detail and its advantages are discussed. To illustrate the use of OpenLearn for QG, a QG system, Ceist [2], is presented which generates questions from OpenLearn content. Ceist is also offered as a case study for how existing NLP tools can be used to build a QG system, and should provide useful information to those considering the development of their own QG system.

Keywords. Question Generation, Data collection, Question Generation System

Introduction

Recently, a number of communities with interest in Question Generation (QG) including those from Natural Language Processing (NLP) and Intelligent Tutoring Systems (ITS) have met with the aim of setting up a Shared Task and Evaluation Challenge (STEC) for QG (http://www.questiongeneration.org).

The current paper aims to contribute to the development of the QG STEC in two ways: firstly by identifying and describing a specific resource for QG, the OpenLearn study units and secondly by demonstrating how state-of-the-art NLP tools can be used and combined to provide a flexible QG system. We explain the process that is used to develop a pattern to match specific target sentences.

The OpenLearn online educational resource project was initiated by the Open University in 2005 with the aim to 'open access to education for all'. Development on OpenLearn began in 2006 and the OpenLearn website now has over 5,400 hours of learning material contained in over 475 study units in a variety of categories from 'Arts and History' to 'IT and Computing' (http://openlearn.open.ac.uk).

A QG system, Ceist, has been created which uses OpenLearn as an input data instance. Newcomers to the QG task and to Natural Language Processing (NLP) will discover that a vast amount of work has already been done to solve many different aspects of NLP [1]. It is hoped that by giving an insight into the methods used by Ceist, others may be inspired to produce their own QG systems.

[1] Corresponding Author: Brendan Wyse; Email: bjwyse@gmail.com
[2] Ceist is pronounced 'kesht' and is the word for question in the Irish language.

1. OpenLearn: A structured online resource of educational materials

There are significant advantages to be gained by using OpenLearn as a data source for QG system development. Some of these advantages are listed below and then described in more detail.

- Applying QG to educational material showcases an excellent practical application of the QG task;
- The material is available in several downloadable formats;
- The material covers a wide range of categories and varying discourse types and, unlike for example Wikipedia articles, it has been authored exclusively by subject experts and passed through a rigorous process of proofreading and editing (with the study units being based mostly on printed Open University course materials);
- The material is available under a Creative Commons 'Attribution; Non-commercial; Share Alike' license.

The ability to enhance the learning process has been identified as a key benefit of QG [2]. QG can be used to automatically create questions for assessment and also provide students with examples of questions as a way to help them learn to formulate their own questions. Research has shown that asking the right questions is a good way to measure how well a subject has been understood following learning. Applying QG to OpenLearn helps to establish a baseline against which future QG systems can be measured and demonstrates the benefits of QG to parties not involved in the area of NLP but in other areas such as education and teaching.

Each OpenLearn study unit is available for download in 8 different formats. The formats include a complete printable article with images, a zip file containing the HTML for the article and its associated images, a RSS XML format, an XML format, a complete zipped set of XML files and images and also the article in formats suitable for use with Moodle[3] and the Common Cartridge[4] specification.

The XML schema used by OpenLearn is well designed. Containing over 1,250 lines it defines a hierarchy within an article that allows an appropriately designed machine to easily retrieve the sections of the article it needs. It permits subsections within sections and different types of content within each section such as the section title, paragraphs, media content and exercise activities as can be seen in Fig. 1. This organization of content facilitates the writing of a script for selecting only the relevant parts from each section, ignoring titles or media content.

[3] A web application used to create online learning sites. See http://moodle.org
[4] A package interchange format for learning content. See http://www.imsglobal.org

```
<SubSection id="SEC001_002_003">
    <Title>Fall of the Bastille, 14 July 1789</Title>
    <Paragraph>In a similar mood of aggrieved self-righteousness and re
    <Paragraph>The event seemed to its supporters literally epoch-makin
    <Paragraph>In France, the anniversary of the taking of the Bastille
    <MediaContent src="Gustav 3" id="PDF001_002" type="pdf" target="new
        <Caption/>
        <!-- optional  -->
        <SourceReference/>
        <!-- Optional  -->
        <Description>Click on 'View document' to read Gustav III of Swe
        <!-- mandatory  -->
    </MediaContent>
    <Activity id="EXE001_002">
        <Heading>Exercise</Heading>
        <Question>
            <Paragraph>Now read the second document (letter from Gustav
```

Figure 1. An example of OpenLearn's XML format.

The fact that the range of topics covered is so varied and the articles are available for research makes OpenLearn an ideal data instance for QG system development and testing. Fig. 2 below shows the process for taking an OpenLearn article and generating the input data for Ceist. In the next section, the process of generating questions from this input with Ceist is described.

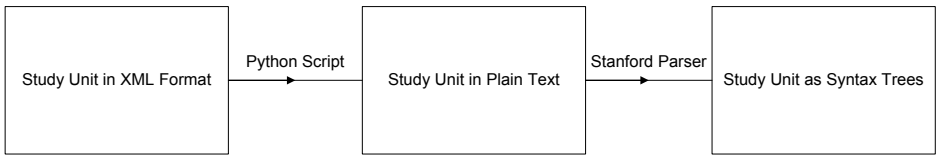

Figure 2. Preparing an OpenLearn study unit for Ceist.

2. Ceist v0.1: A Question Generation system

Many of the current documented QG systems use pattern matching to find sentences for which pre-determined questions can be generated based on rules [3][4][5]. This technique was also used with great success in the QA task [6]. Ceist v0.1 follows the same approach.

Ceist takes syntax trees for its input and uses a pattern as part of a rule, to match sentences which are compatible with specific rules. Ceist then uses templates in conjunction with the matched parts of the sentence, to generate a question and the corresponding (short) answer. The main focus for Ceist was to allow the sentence matching patterns in the rules to have maximum flexibility: the patterns can be written to match a very wide range of word sequences or a very specific and narrow range.

As shown in the System Architecture diagram (Fig. 3), Ceist uses the Tregex [7] tree searching code provided by the Stanford NLP group for pattern matching. Ceist achieves flexibility by allowing patterns to be defined to match an individual word or regular expression, up to a POS tag or group of POS tags such as a Noun Phrase. This means all input data must be parsed into a syntax tree during pre-processing. Ceist then has the power to match any individual node in that tree or any specific sub-tree regardless of the node value being text, a POS tag or a phrase grouping tag.

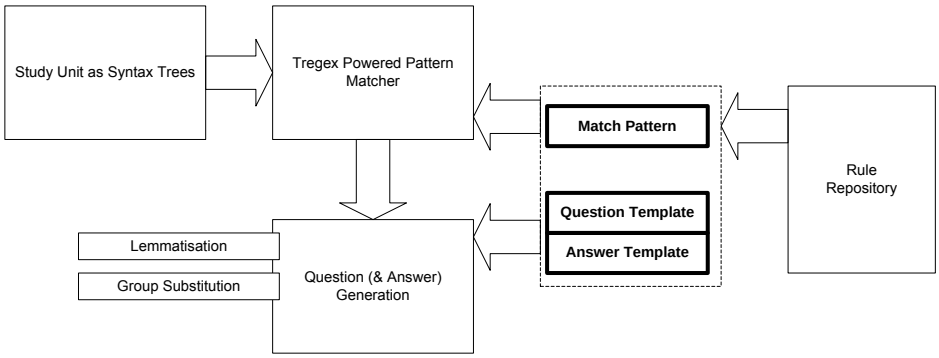

Figure 3. Ceist v0.1 architecture.

The rest of this section will describe the process of customising Ceist such that it can recognise a sentence from an OpenLearn article about the French Revolution and generate a question for that sentence. The sentence which will be the 'target' for the rule is *'Emmanuel-Joseph Sieyès (1748-1836) trained as a priest and became assistant to a bishop'*. The rule will be written so as to generate the question *'What did Emmanuel-Joseph Sieyès train as?'*.

The steps commonly taken to allow a machine to manipulate or find patterns in words are to tag those words in a sentence and then optionally to group the words into a hierarchy of their sentence parts. Words can be nouns, verbs (in various tenses), and adjectives amongst other types. They can also be grouped into phrases such as noun phrases or verb phrases. Sentences may also contain cardinal numbers and special symbols such as parentheses or brackets, just like the dates in our target sentence, *'(1748-1836)'*.

The manner in which sentences are parsed into their parts is a well researched area and the state-of-the-art has reached a level of performance which provides excellent results for most sentences [8]. As shown in Fig. 2, the Stanford parser (http://nlp.stanford.edu/software/lex-parser.shtml) is used for parsing. It is applied to the relevant text extracted from the XML formatted version of the article as seen in Fig 4. The parser provides the parsed output as a syntax tree as shown in Fig. 5 using labelled bracketing notation.

```
            <SubSection id="SEC001_002_002">
                <Title>The Third Estate as the voice of the nation</Title>
                <Paragraph>
Emmanuel-Joseph Siey&#xE8;s (1748&#x2013;1836) trained as a priest and became
assistant to a bishop. He had no religious vocation, however, and his fame arose as
the author of a highly influential pamphlet, <i>What is the Third Estate?</i>,
```

Figure 4. An OpenLearn article in the XML format.

```
(ROOT (S (NP (NP (NNP Emmanuel-Joseph) (NNP Sieyès)) (PRN (-LRB- -LRB-) (NP (NP
(CD 1748)) (: --) (NP (CD 1836))) (-RRB- -RRB-))) (VP (VP (VBN trained) (PP (IN
as) (NP (DT a) (NN priest)))) (CC and) (VP (VBD became) (NP (NN assistant)) (PP
(TO to) (NP (DT a) (NN bishop))))) (. .)))
```

Figure 5. Syntax Tree output provided by the Stanford parser.

The syntax tree in Fig. 5 can be displayed in a more discernable view using source code again made available by the Stanford NLP team and implemented in Ceist as a feature. Ceist was designed to assist rule design by providing features such as the syntax tree display shown in Fig. 6. This visual aid provides a reference for rule authors and helps them to derive match patterns using the tree nodes.

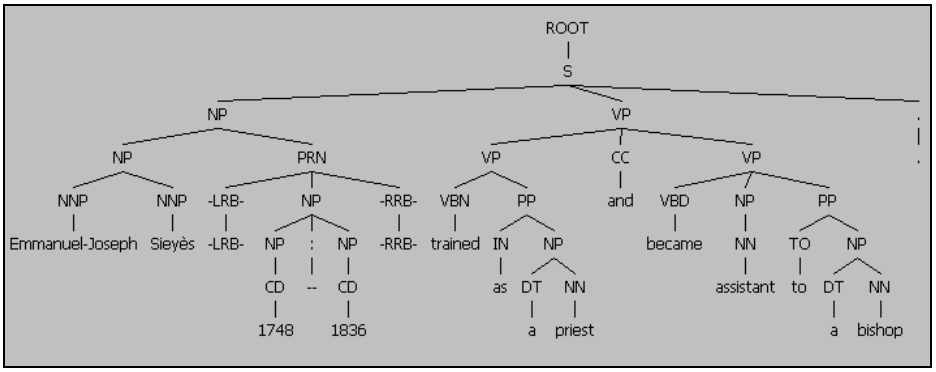

Figure 6. The Syntax Tree as displayed in the QG application.

The tree in Fig. 6 has for its leaves, each word of the sentence. The parent node for each leaf is the Part-Of-Speech (POS) tag. A noun is represented by NN, a proper noun by NNP. The representation used depends on the parser used. The POS tags are all children of a group which can be a noun phrase (NP) or a verb phrase (VP) among others. Given the parsed input sentence, as shown in Fig. 6, we would like to write a pattern which matches this sentence.

The flexibility of Tregex was described earlier. We can write a very narrow focused pattern to only match the exact sentence above using the example shown in Fig. 7.

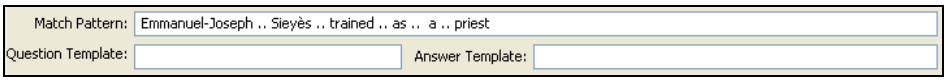

Figure 7. A match pattern with low coverage due to narrow expression terms.

This pattern contains exact words which must be matched and the double dots indicate that the words must be in sequence (but not necessarily immediately preceding one another). It is a valid pattern and will match our target sentence, but the coverage is low. The pattern will only ever match very specific sentences which contain the words from the pattern in the correct order. It would be better if this rule could cover not just *Emmanuel-Joseph Sieyès* but any priest. It would be even better if the rule was not restricted to just priests!

Referring back to the syntax tree in Fig. 6, we can use the groups and POS tags to give our rule a bit more coverage. The rule is rewritten to match a noun phrase (NP), followed by a verb in the past participle (VBN), the words 'as a' and a noun (NN). In Fig. 6 one can see that two noun phrases are nested on the left most branch containing the persons name. To ensure we match the lowest noun phrase in the branch, we use the syntax '!< NP' meaning does not have a child which is a noun phrase.

At this point we will also mark those parts of the match which we are interested in for generating our question, the initial noun phrase and the verb. We need to use those to generate the question and answer.

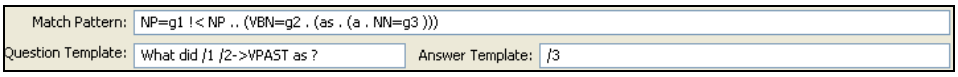

Figure 8. A pattern with high coverage through use of broader expression terms.

The pattern in Fig. 8 has a much wider coverage. It also marks the matched parts using the syntax '=gX' where the value X is then referenced in the question template. The question template uses the syntax '/X' to indicate where the matched parts are inserted to generate the question.

One NLP tool which can be used to change verb tenses is a morphology database. The advantage of such a tool is that each word has been manually entered into the database and therefore it is very accurate. The disadvantage is that new words must be added to the database over time. The Xtag morphology database [9] was ported to Java to provide this feature in Ceist. A sample line from the flat file version of the morphology database contains a keyword followed by other forms of that keyword:

```
trained              train   V PAST WK#train       V PPART WK
```

The database is queried by sending a keyword and a command. The XTAG program then finds the entry in the database for that keyword and looks up the word for which the inflected form is the given command. For example, the VPAST WK of 'train' is 'trained'. If we send the query 'trained' and the command 'VPAST', the database returns 'train'. This allows us to change the form of the verb when we generate the question by using the '->VPAST' modifier.

Higher coverage can, however, also introduce problems: whenever the coverage of a pattern is widened the risk of matching unwanted sentences increases. This can be seen when the rule above is applied to the entire article on the French Revolution; the output for this is shown in Fig. 9. Another feature to aid rule editing with Ceist is the rule output. This updates instantly when the match pattern is edited; note how Ceist also marks those parts of the target sentence which will be used in the question and answer templates with superscripted integers and colouring.

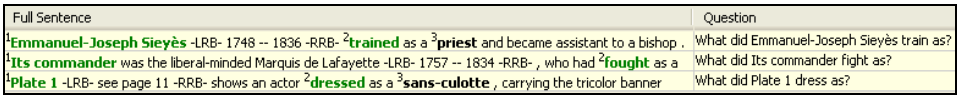

Figure 9. Results display showing all matched sentences and generated questions.

The pattern matches three sentences from the OpenLearn article. The three sentences which were matched cannot be seen completely in Fig, 9, but they are in full:

1. Emmanuel-Joseph Sieyès (1748–1836) trained as a priest and became assistant to a bishop.

2. Its commander was the liberal-minded Marquis de Lafayette (1757–1834), who had fought as a volunteer with the American revolutionaries.

3. Plate 1 (see page 11) shows an actor dressed as a *sans-culotte*, carrying the tricolour banner (on which is emblazoned the slogan liberty or death') at the 'festival of liberty' in Savoy in October 1792.

Now that the coverage has been broadened, the pattern author must decide which of these sentences they are attempting to target. The matches might produce valid questions, but if a pattern is intended to produce other questions such as 'What was the occupation of Emmanuel-Joseph Sieyès', then it might be better to refine the pattern further. To achieve this the rule author must refine the rule.

One distinguishing factor evident from the results is that the first matching part is a person's name for our target sentence, but not for the others. Another is the fact that the subject of the target sentence is the noun phrase immediately preceding the verb. This is important because it is quite possible that a sentence very similar to sentence 3 would be incorrectly matched if the first noun phrase were a persons name.

Currently, rules are manually refined by modifying the pattern to exclude irrelevant sentences. This is a slow process but does result in improved rules over time.

The expression 'NP <- NNP !<, DT' matches a noun phrase which does not begin with a determiner and ends in a proper noun. This expression matches person names and is useful for matching sentences containing person names. Ceist provides the capability to use a definition for a person name which is then substituted for this expression whenever a pattern seeks to match a noun phrase which is a person name.

The 'NP' in the original pattern can be replaced with 'personNP' and the rule will then only match a noun phrase which is a person's name. Ceist also allows this technique to be used for groups such as colours or the days of the week. One advantage of this technique is that the definition for personNP can be refined in one place, without needing to rewrite all rules containing personNP.

Tregex allows us to specify that a noun phrase must immediately precede the verb by using a single dot instead of the double dots we were using.

The refined pattern now looks like that in Fig. 10. When using groups such as 'personNP', the reference number does not use the syntax '=gX', but instead the reference number is appended to the group name.

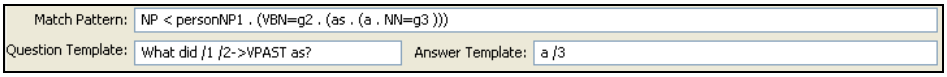

Figure 10. Modified pattern which matches a persons name.

This new pattern eliminates the undesired results returning a single match and generating the question we originally targeted. Fig. 11 shows the final result.

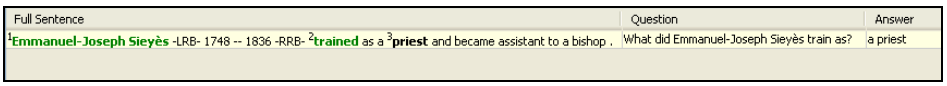

Figure 11. Result showing the desired match and generated question.

The approach used by Ceist is similar to that used by many existing Natural Language Generation (NLG) systems. Rules consisting of patterns and templates are defined, with the template reusing some of the words from the input that matched the pattern. The manner in which the rules are represented does vary however.

Ceist stores its rules as XML and uses a format similar to the QuALiM Question Answering system [6]. Matched parts of the sentence are marked as integers and the templates reference these integers.

Cai et al. also use a format which they present as a mark up language, NLGML [5]. NLGML marks matched parts of the original sentence as variable names for use in the templates and where Ceist replaces semantic features such as a persons name with a new match type, NLGML uses attributes within the noun phrase's XML tags.

3. Conclusions and Further Work

This paper introduced OpenLearn as a suitable data resource for Question Generation and described the Ceist Question Generation system. Ceist uses a rule-based approach based on pattern-template combinations and provides several facilities for making the rule authoring process more user-friendly.

We would like to conclude by proposing that one way to advance the state-of-the-art in QG is to initiate an effort to arrive at a standard format for the representation of rules. Ideally rules should be shareable and possibly even come with information on their coverage and error rates. The main problem to overcome will be to make sure that a common format does not commit the QG community to a restricted set of NLP tools.

References

[1] D. Jurafsky and J.H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*, Prentice-Hall, New Jersey, 2008.

[2] A. Graesser, J. Otero, A. Corbett, D. Flickinger, A. Joshi and L. Vanderwende, Chapter 1: Guidelines For Question Generation Shared Task Evaluation Campaigns, In V. Rus and A.C. Graesser (Eds.) *The Question Generation Shared Task and Evaluation Challenge* (2009), http://www.questiongeneration.org.

[3] D. Gates, Generating Look-Back Strategy Questions from Expository Texts, *1st Workshop on the Question Generation Shared Task and Evaluation Challenge*, NSF, Arlington, VA (2008).

[4] W. Wang, T. Hao and W. Liu, Automatic Question Generation for Learning Evaluation in Medicine, *Advances in Web Based Learning – ICWL 2007* (2008), 242-251.

[5] Z. Cai, V. Rus, H.J. Kim, S.C. Susarla, P. Karnam and A.C. Graesser, NLGML: A Markup Language for Question Generation, *Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education* (2006), 2747-2752.

[6] M. Kaisser and T. Becker, Question Answering by Searching Large Corpora with Linguistic Methods , *Proceedings of the 2004 Edition of the Text REtrieval Conference*, Gaithersburg, Maryland (2004).

[7] R. Levy and G. Andrew, Tregex and Tsurgeon: tools for querying and manipulating tree data structures, *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*, Genoa, Italy, (2006).

[8] D. Klein and C.D. Manning, Accurate Unlexicalized Parsing, *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, Sapporo, Japan, (2003), 423-430.

[9] C. Doran, D. Egedi, B.A. Hockey, B. Srinivas and M. Zaidel, XTAG System – A Wide Coverage Grammar for English, *Proceedings of the 15th International Conference on Computational Linguistics*, Kyoto, Japan, (1994), 922-928.