Learning from student responses: A domain-independent natural language tutor

Jenny McDonald[1], **Alistair Knott**[2], **Richard Zeng**[1] **and Ayelet Cohen**[1]
[1]Higher Education Development Centre, University of Otago
[2]Dept of Computer Science, University of Otago
jenny.mcdonald@otago.ac.nz

Abstract

Providing timely and individualised feedback to students in large undergraduate classes is problematic. In this paper we describe our approach to creating a simple, surface-based, domain-independent natural language tutor which uses simple machine learning techniques as a step towards resolving this issue. The focus of our efforts was on developing a high-quality tutorial dialogue plan, creating well-designed questions, and building a model of student responses derived from real student data. We present some early evaluation results and briefly outline the opportunities that our approach and the new tutorial dialogue system present.

1 Introduction

The study this paper describes arose out of a practical need expressed by one of the coordinators of an undergraduate first year health sciences course:

> For large class sizes of 1500-1800 students, is it possible to use technology to provide timely individualised feedback to students on their understanding of key concepts?

Natural language intelligent tutoring systems (ITS) seemed to offer some promise for supporting and enhancing student understanding of key concepts in this domain. For example, Circsim Tutor (Evens and Michael, 2006) is a natural language tutor designed expressly to develop health sciences student understanding of the baroreceptor reflex in humans (the baroreceptor reflex is one of the mechanisms for maintaining blood pressure in humans). Nevertheless derision and dismisal of ITS as a failed enterprise are common views among many educational researchers and practising teachers, for example, Laurillard (2002) and Ramsden (2003). With a few exceptions, even today, within Higher Education, ITS are hardly in widespread practical use for teaching and learning (Reeves and Hedberg, 2003). There are some good practical reasons for this. Murray (1999), in his review of ITS authoring systems, addresses a key one:

> Building an explicit model of anything is not an easy task, and requires analysis, synthesis, and abstraction skills along with a healthy dose of creativity. ...it is difficult to reduce the entire design task to low level decisions that yield a quality product...some degree of holistic understanding and abstract thinking will eventually have to come into play.

Worse still, in practice it is seldom feasible to adapt a system designed for a specific teaching and learning context to another. So for example, while Circsim Tutor deals with the baroreceptor reflex it deals with it at a level which is too advanced for the broader introductory-level course on cardiovascular homeostasis that we were dealing with. Even if this were not the case, there would likely be differences in emphasis in terms of the curriculum and adapting a deep system like Circsim would be a non-trivial task.

As the authors' primary focus is on teaching and learning development across a tertiary institution, we required a system that would be both responsive and practical in real class settings and which could be readily adapted to a wide range of domains. We decided to build a very simple, surface-based, domain-independent natural language tutor using simple machine learning techniques, and to focus our efforts on developing a high-quality *lesson plan*. The lesson plan should have two components. Firstly, there should be a well-designed set of questions for the tutor to ask, to probe students' knowledge of the domain. Secondly, for each question there should be a good model of the range of common answers which students are likely to provide, so that the tutor can give suitable responses to each of these, and give students the individualised feedback which is the key goal of the system.

This development method hinges firstly on in-depth interactions with the teaching staff of the course, and secondly on the acquisition of high-quality training data from actual students. Consequently, there were two stages in the development of the system. In the first stage, we produced a detailed set of questions, in close liaison with the teaching staff, and devised a detailed script, providing for possible student responses to each question, along with appropriate tutor actions, and then used an existing surface-based dialogue engine to put these questions to students and give responses based on simple pattern-matching techniques. This stage-one system was mainly intended as a means for gathering training data for the full system, though of course we intended it to have some educational merit as well. In the second stage, we refined the script we had developed in the first stage to add new response categories, manually classified student responses to each question, trained a set of classifiers on the revised set of categories and developed a second dialogue manager which deployed these classifiers.

Of the few readily available domain-independent tutorial dialogue systems, TuTalk from the Learning Research and Development Centre at the University of Pittsburgh, is one that provides a relatively easy way to author dialogues using only a text editor and scripts can be authored much like writing a natural one-to-one dialogue. The TuTalk scripting language represents each tutor turn as a finite state machine (FSM) and the author can define multiple classes of student response that lead from the current tutor turn to the next tutor turn or state. A simple chat-style interface is all that is required for a user to interact with the system (Jordan, 2007).

However, TuTalk 'out of the box' includes many features and options that we did not need and added processing overhead to response times especially for longer responses. We were also aware that the basic natural language understanding module that came with it was fairly rudimentary and yielded poor precision and recall when classifying most student responses other than those to simple binary or limited-option questions. Nevertheless, with the addition of a web interface, a large cohort of students could potentially access and use the system, and early feedback from students indicated that even though their responses were often not well 'understood' they found the dialogue useful as a study and revision tool (the default position was always to provide a model answer): for these reasons we felt that TuTalk would allow us to rapidly acquire a large corpus of authentic student responses to questions posed in the dialogue. Once we had captured this data we could then proceed to build a very simple, surface-based natural language dialogue system which retained a FSM model for representing the dialogue. The large corpus acquired via the TuTalk system would be used to both train the new system and to inform revision of the script itself.

In the next section of this paper we describe collecting the dialogue corpus and evaluating performance. The third section outlines the new dialogue system architecture and describes how we have utilised components from the Natural Language Toolkit (Bird, 2006). The fourth section describes the iterative process we are using to both develop the script and build a set of classifiers for student responses based on insights and intuitions from real student data. In the fifth section we present some early classifier test results and in the final section we discuss the educational research opportunities that both our approach and new tutorial dialogue system present.

Dialogue excerpt
TUTOR: Would you like more practice with the ideas we've discussed so far, before we move on? STUDENT: yes please TUTOR: If heart rate drops and stroke volume remains the same, what will happen to cardiac output? STUDENT: decrease TUTOR: Well done. Cardiac Output is determined in part by Stroke Volume. How do you calculate CO? STUDENT: co = heart rate x stroke volume TUTOR: Cool, that's right. If cardiac output increases and TPR remains the same what will happen to mean arterial pressure? STUDENT: increase

Table 1: Tutorial dialogue excerpt

2 Collecting the dialogue corpus

In close consultation with the course lecturers an initial tutorial script covering the curriculum on cardiovascular homeostasis was developed. The script was written by a medical graduate using lecture notes, laboratory manuals and self-directed learning material from the course and required approval from the teaching team before it could be deployed for students. The teaching team wanted to be certain the script contained no errors of fact and that it had some educational merit. Table 1 shows an excerpt from a tutorial dialogue session. The script was refined by incorporating utterances, captured from pilot dialogue interactions between the system and student and staff volunteers (N=34), back into the script. Incorrect behaviour of the script was dealt with through adjustments to tutor turns and improvements to script flow was made following feedback from students and tutors on the course and from examination of the dialogue transcripts. Feedback from students and some of the teaching staff indicated to us that the system even without much in the way of 'understanding' still had some value as a learning and teaching tool because it always provided model answers to questions.

2.1 Student responses

The cardiovascular homeostasis tutorial was released to the first year undergraduate class at the beginning of their module on the human cardiovascular system. Tutorial use was optional. 437 students accessed the system during the course (total class enrolment=1800) and produced a total of 532 dialogues; several students accessed the dialogue more than once. However from the total number of dialogues, only 242 dialogues were completed through to the half-way point and only 127 dialogues were completed to the end. A handful of dialogues were interrupted because of system-related problems but the majority that terminated before completion did so because because the students simply ended their session. Feedback from course tutors and comments from the students themselves supported our inuition that poor system 'understanding' of student dialogue contributions was probably a key reason for the fall-off in use. Nevertheless, it served its purpose in capturing a large quantity of training data which is mainly what it was for.

3 Dialogue system architecture

Clearly we needed to improve the 'understanding' performance of the dialogue system if we were to hope to provide individualised feedback on free text input: two options were considered. Either we could continue to use TuTalk and replace the existing TuTalk natural language understanding (NLU) module along with making adjustments to the script design and dialogue manager (DM) or we could build another dialogue system from scratch. In the end we chose to start from scratch for three main reasons. First, the natural language toolkit (NLTK) already provided many of the functions required in a simple dialogue system such as tokenisers, stemmers and a range of classifiers. Second, for our purposes, we didn't require many of the features built into TuTalk and we had experienced some performance issues with the system. Third, a very simple modular system that could be easily extended or adapted, and which utilised well established libraries would provide a solid base from which to do further work in this area. Fig. 1 provides an overview of our system architecture.

The dialogue system is written in python and

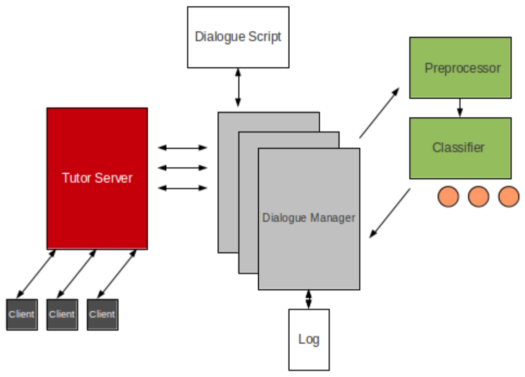


Figure 1: Architecture of Dialogue System.

utilises several NLTK libraries, Peter Norvig's 'toy' spell checker, and the Asyncore and Asynchat libraries to manage multiple simultaneous client connections. The server can readily communicate with any web-application front end using XML-formatted messages. We have also built a java-based web application through which multiple clients can connect to the tutorial server (Fig. 2).

The structure of the tutorial dialogue is determined entirely by the dialogue script. We wanted to use a FSM model for the dialogue since this permits an organic authoring process and there is no theoretical limit to how deep or broad the dialogue becomes.

The script structure itself is based on Core and Allen's (1997) dialogue coding scheme and each dialogue contribution is divided into forward and backward functional layers. Jumping to alternative parts of the script is embedded in the forward function rather than being treated as a separate layer. This seems to work well with the notion of 'action-directive' functions proposed by Core and Allen. In effect the forward functions always advance the dialogue even if some elements are repeated along the way. It also seems like a more intuitive and less confusing approach than incorporating special tags in either the backward layer, or inventing a new layer to handle them. The script is an XML file which is defined in our XML schema for the dialogue system and which essentially comprises a series of dialogue contributions.

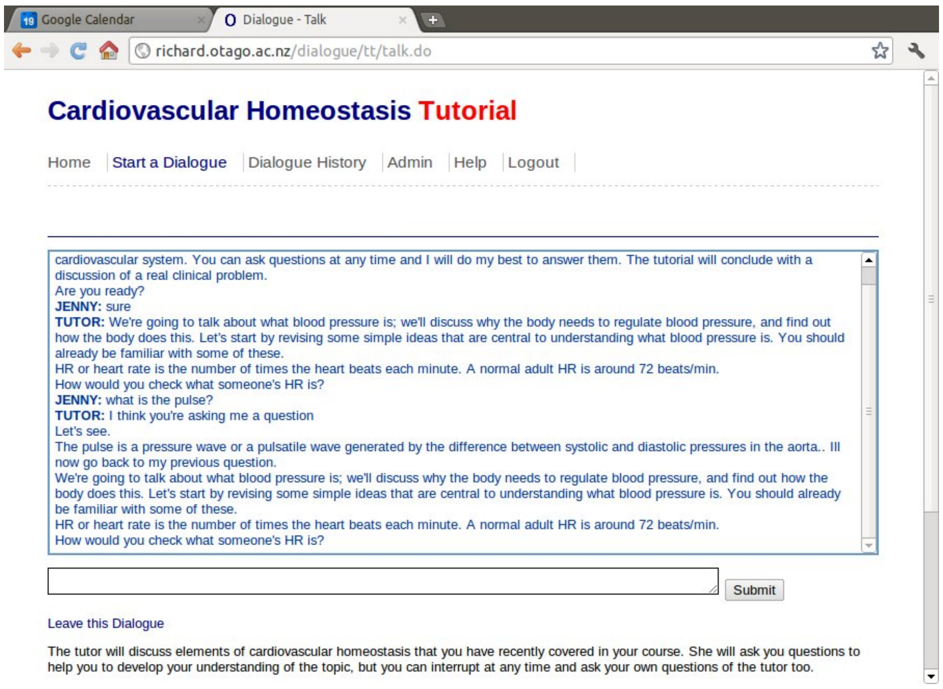
An example of a single dialogue contribution, called a contribution node is given in Figure 3. In this example, the unique id of the dialogue contribution is "check-hr". Apart from the start and end nodes of the dialogue, every contribution node has a backward and forward layer. The backward layers contains responses appropriate to the previous dialogue context, for example an utterance to establish grounding (Clark and Schaefer, 1989), and the forward layer sets up the next dialogue context.

```
<contribution-node id="check-hr"
parent-node="start"
default="true">

<backward class="yes">
<acknowledge/>
</backward>

<forward>
<assert>We're going to talk about
what blood pressure is; we'll discuss
why the body needs to regulate blood
pressure, and find out how the body
does this. Let's start by revising
some simple ideas that are central to
understanding what blood pressure is.
You should already be familiar with
some of these.</assert>

<assert>HR or heart rate is the number
of times the heart beats each minute.
A normal adult HR is around
72 beats/min.
</assert>

<info-request value="How would you
check what someone&apos;s HR is?"
define="You could take their
pulse."/>

</forward>
</contribution-node>
```

Figure 3: *check-hr* contribution node

While the tutorial system is primarily designed for single-initiative dialogue, the opportunity for limited mixed-initiative is incorporated through classifying question contributions at any stage of the dialogue and searching for possible answers within the dialogue script. In addition, the script can be designed to accomodate opportunities for eliciting further explanation where the need is apparent from examination of previous student responses. (For an explanation of this see Section 4).

Each client connection to the system creates an



cardiovascular system. You can ask questions at any time and I will do my best to answer them. The tutorial will conclude with a discussion of a real clinical problem.
Are you ready?
JENNY: sure
TUTOR: We're going to talk about what blood pressure is; we'll discuss why the body needs to regulate blood pressure, and find out how the body does this. Let's start by revising some simple ideas that are central to understanding what blood pressure is. You should already be familiar with some of these.
HR or heart rate is the number of times the heart beats each minute. A normal adult HR is around 72 beats/min.
How would you check what someone's HR is?
JENNY: what is the pulse?
TUTOR: I think you're asking me a question
Let's see.
The pulse is a pressure wave or a pulsatile wave generated by the difference between systolic and diastolic pressures in the aorta.. Ill now go back to my previous question.
We're going to talk about what blood pressure is; we'll discuss why the body needs to regulate blood pressure, and find out how the body does this. Let's start by revising some simple ideas that are central to understanding what blood pressure is. You should already be familiar with some of these.
HR or heart rate is the number of times the heart beats each minute. A normal adult HR is around 72 beats/min.
How would you check what someone's HR is?

The tutor will discuss elements of cardiovascular homeostasis that you have recently covered in your course. She will ask you questions to help you to develop your understanding of the topic, but you can interrupt at any time and ask your own questions of the tutor too.

Figure 2: Screenshot of Dialogue System Web Client.

instance of the dialogue manager which sends tutor contributions to the client according to the preloaded script and receives student contributions which are then classified and determine the next tutor contribution. The pseudocode given in Fig. 4 illustrates how the dialogue manager processes each contribution node.

The dialogue system is intended to be domain neutral since the content and structure of the dialogue is determined solely by the dialogue script (note that we have yet to demonstrate this by building scripts outside the domain discussed in this paper). Scripts are designed according to the XML schema specified for the dialogue system. A domain-appropriate dictionary is required for the spell checker: for our cardiovascular homeostasis tutorial we combined the text from pilot student responses, the script itself, the relevant section from an accessible human physiology text (`http://en.wikibooks.org/wiki/Human_Physiology/The_cardiovascular_system`) and the NLTK plain text ABC science corpus.

4 Building classifiers and revising the script

Tutorial dialogue script revision and classifier development are currently underway. The approach we have taken is to do these two tasks hand-in-hand. In this section we describe the rationale for this approach by way of illustrative examples.

In general, a separate classifier is required for each dialogue contribution in the script. So for example, the dialogue contribution *check-hr* has its own classifier. In this case, one of the possible classes for text classified is *correct-simple* and this is specified in the backward class attribute value.

Each backward layer must have a class attribute. When the previous student dialogue contribution matches this class, this contribution node becomes the current node. In the example above, responses to the parent contribution node *check-hr* are processed by a single classifier into one of the classes listed in Table 2. If classification fails then the dialogue contribution which is specified as the default is chosen.

The process of building a classifier for each dialogue contribution requires a number of steps. First, classification by hand of a training set derived

```
If there is user input
=>Normalise and spell-check raw input
=>Select appropriate classifier
for current dialogue contribution
based on dc label
=>Get next dialogue contribution
node from classifier result

If dialogue contribution has a backward
layer
=>send contents to client

If dialogue contribution has a
forward layer
=>send any assertions to client
=>send info requests to client

if action is required (for
example jump to a specific dialogue
contribution or question answering
routine)
=>go to required node
```

Figure 4: Dialogue manager pseudocode

check-hr classifier
correct
correct-simpler
question
incomplete

Table 2: Possible classes for check-hr

```
<contribution-node id="hr-simpler"
parent-node="check-hr">

<backward class="correct-simpler">
<acknowledge/>
<part-agree/>
</backward>

<forward>
<info-request value="Can you
think of a simpler method?"
action="check-hr"/>
</forward>

</contribution-node>
```

Figure 5: *hr-simpler* contribution node

from the student corpus. The XML schema of the NPSChat corpus provided with the NLTK was a useful model for us to follow in marking-up the corpus and allowed us to use the appropriate NLTK corpus reader directly. The classes used are created based on inspection of student responses although the class *question* and the class *correct* are used in each classifier. For example, examination of student responses to the question:

> How would you check what someone's HR is?

led to us developing a class *correct-simpler* given that a handful of students suggested using an ecg or blood-pressure cuff and stethoscope. These methods are not the easiest ways to achieve this but they are valid answers and lend themselves to seeking a simpler method in order to check student understanding.

This process in turn may require a new dialogue contribution for the script. For example, the contribution node *hr-simpler* (Fig. 5.) was only created after the classifier for *check-hr* had been built and then became an addition to the original script. This is why we suggest script revision and building of classifiers should be done together.

For each dialogue context a training set is created. Typically the first 100 student responses for each tutor question are classified by a human marker, although this number may be less where it is clear that there is little variabiliy in student responses (for example, in the case of binary questions) or more where there is a wide range of student responses. Once a suitable training set is marked up the set is divided into 5 folds and a Naive Bayes classifier trained on 4/5 folds initially using simple *bag of words* as the featureset and then tested on the remaining fold. A 5-way cross-validation is carried out and accuracies for each of the 5 test sets calculated. The average accuracy across the 5 test sets and standard deviation is also recorded.

This process is then repeated using different featuresets (for example, bag of words, word length, first word, with/without stemming, with/without stopwords etc) until the highest accuracy and least variability in test set results is achieved. Some features are particularly appropriate in a given context. For example, length of response is a good predictor of an incomplete answer in the *check-hr* context above. A student common response in this context was simply 'pulse' and the human classifier had de-

Question Type	**Example**
binary	'Are you ready?'
limited-option	'How would you check somone's heart-rate?'
open	'What is the pulse?'

Table 3: Question Types

cided these responses were incomplete and required, 'count' or 'measure' or a similar qualifier before accepting this as a correct answer. Response length helped to distinguish these responses from correct responses.

Once the best features for a given dialogue-contribution classifier have been established the size of the training set is increased in order to expose the classifier to a larger number of samples and improve accuracy. Finally, where uncommon but pedagogically useful student responses are found, the training data may be weighted with these in order to increase the likelyhood that similar responses are correctly classified.

The classifier is evaluated with previously unseen data and scored relative to a human marker. The entropy of the probability distribution (E) is calculated for each unseen response and this is used to determine appropriate thresholds for classification. For example, if E is close to zero the classifier confidence is generally very high. E 1 indicates low confidence and less difference between the class rankings.

Finally the classifier is serialised, along with its associated feaureset parameters and saved for use in the dialogue system itself.

5 Testing classifiers

In general there are three types of tutor question in our cardiovascular homeostasis dialogue: binary, limited-option and open. An example of each is given in Table 3.

Evaluation of classifiers for binary questions has resulted in the highest accuracy with the smallest amount of training data (98-99 percent on training set size of 200). Typically, for this question type, there are only two class options plus a third to cater for a question initiative from the student. In this section we focus on classifying responses for the remaining two question types, limited-option and open, since these tend to be more educationally interesting and relevant, and harder to classify. Data is presented for an example of each type of question.

1. Limited-option. The *check-hr* dialogue contribution is a good example. Even with free text, there is a reasonably limited number of ways to answer the question, 'How would you check someone's heart-rate?'. Indeed the great majority of student responses were of the form *count the pulse, measure the pulse, take the pulse, etc*. Best results were achieved using a combination of the NLTK Porter stemmer on tokenised words, word length, first word, and a custom regular expression feature to pick up reference to ECG or blood pressure. Using these features, accuracy increased from a best of 0.72 to 0.90 when training data increased from a fold size of 19 to a fold size of 204 (Refer Fig. 3). Variability in training set accuracy was reduced when stopwords were removed from less than 0.03 to less than 0.01.

Evaluation on the trained classifier on a previously unseen sample of 20 was surprisingly 100 percent with entropy values between 0.00 for a student response, 'by measuring their pulse rate' to 0.81 for 'by listening to their pulse'.

2. Open. A good example in this case, is the question, 'What is the pulse?'. There is a wide range of ways in which an answer to this question could be reasonably expressed. The model answer given is 'The pulse is a pressure wave or a pulsatile wave generated by the difference between systolic and diastolic pressures in the aorta.' To give an idea of how open this type of question is, the following is an alternate but valid expression of the same idea, 'The pulse is generated by contraction of the heart during systole and is transmitted as a wave to the peripheral arteries.'

Similar to the first case the average accuracy of the classifier we created for this question plateaued at 0.89 with a training data fold size of 194, however it performed far worse on smaller training sets achieving an average accuracy of only 0.41 with a training data fold size of 19. The most useful features in this case were word stems, word length and first word. We had a total of 8 classes as there was a wider range of student responses to the question.

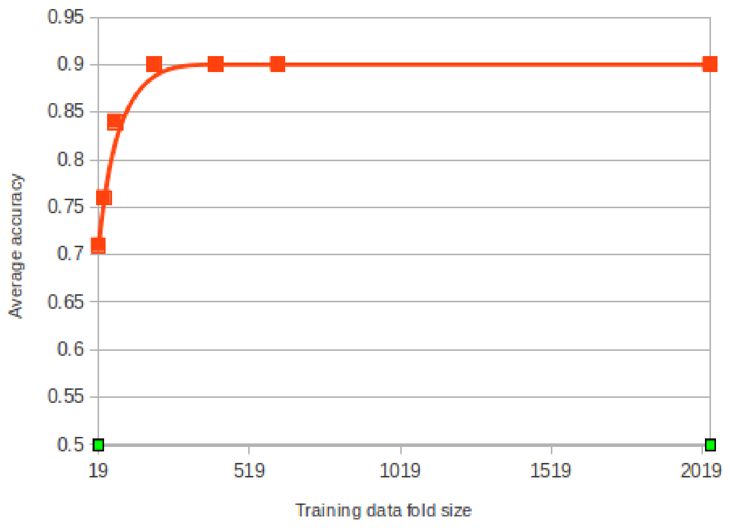


Figure 6: check-hr Training data. Fold size vs Average accuracy

Two of these classes were needed to deal with commonly occuring misconceptions. However one of the classes is redundant. The *incorrect* class regularly failed to correctly identify incorrect answers. The reason for this is likely to be the high degree of variability in incorrect answers unless they demonstrate a commonly held misconception. We expect to achieve better results on our unseen evaluation data after removing the incorrect class from the training set.

6 Discussion

Our goal was to build a very simple, surface-based, domain-independent natural language tutor using simple machine learning techniques, and to focus our efforts on developing a high-quality lesson plan, so that the tutor can ask well-designed questions, and has a good model of the range of possible answers which students will provide for these. Our development method requires in-depth interactions with the teaching staff of the course, plus the acquisition of high-quality training data from actual students.

In this paper we have focussed particularly on describing the system and reporting our approach to building classifiers and script revision in order to achieve this goal. The results of our early classifier evaluations look promising in terms of the ability of the system to 'understand' student responses and take appropriate action. Our next task is to evaluate our system and revised tutorial script with a new cohort of first-year health sciences students. We also plan to compare student learning outcomes against the same script using a multi-choice selection rather than free text responses. Previous investigations in this area have produced equivocal results. For example, Corbett et al. (2006).

We see potential for our approach and the system in a number of areas: supporting the rapid capture of tutorial corpora across a range of subject domains, developing faster and more flexible approaches to authoring tutorial dialogues and of course we hope to make headway with the problem of providing timely and individualised feedback to students which is so keenly sought by our colleagues teaching large undergraduate courses.

References

Steven Bird. 2006. NLTK: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, COLING-ACL '06, pages 69–72, Stroudsburg, PA, USA. Association for Computational Linguistics.

Herbert H. Clark and Edward F. Schaefer. 1989. Contributing to discourse. *Cognitive Science*, 13(2):259–294.

Albert Corbett, Angela Wagner, Sharon Lesgold, Harry Ulrich, and Scott Stevens. 2006. The impact on learning of generating vs. selecting descriptions in analyzing algebra example solutions. In *Proceedings of the 7th international conference on Learning sciences*, ICLS '06, pages 99–105. International Society of the Learning Sciences.

Mark G. Core and James F. Allen. 1997. Coding dialogs with the damsl annotation scheme. In *Working Notes of the AAAI Fall Symposium on Communicative Action in Humans and Machines*, pages 28–35, Cambridge, MA, November.

Martha Evens and Joel Michael. 2006. *One-on-One Tutoring by Humans and Computers*. Lawrence Erlbaum Associates.

Pamela Jordan. 2007. Topic initiative in a simulated peer dialogue agent. In *Artificial Intelligence in Education (AIED 07)*.

Diana Laurillard. 2002. *Rethinking University Teaching*. Routledge Falmer, 2nd edition.

Tom Murray. 1999. Authoring intelligent tutoring systems: An analysis of state of the art. *International Journal of Artificial Intelligence in Education*, 10:98–129.

Paul Ramsden. 2003. *Learning to teach in higher education*. Routledge Falmer, 2nd edition.

Thomas C Reeves and John G Hedberg. 2003. *Interactive learning systems evaluation*. Englewood Cliffs, New Jersey : Educational Technology Publications.