

Minimal Text Structuring to Improve the Generation of Feedback in Intelligent Tutoring Systems

Susan Haller
Computer Science Department
University of Wisconsin–Parkside
Kenosha, WI, 53141, USA
haller@cs.uwp.edu

Barbara Di Eugenio
Computer Science Department
University of Illinois
Chicago, IL, 60607, USA
bdieugen@cs.uic.edu

Abstract

The goal of our work is to improve the Natural Language feedback provided by Intelligent Tutoring Systems. In this paper, we discuss how to make the content presented by one such system more fluent and comprehensible, and we show how we accomplish this by using relatively inexpensive domain-independent text structuring techniques. We show how specific rhetorical relations can be introduced based on the data itself in a bottom-up fashion rather than being planned top-down by the discourse planner.

Introduction

Intelligent Tutoring Systems (ITSs) help students master a certain topic. Research on the next generation of ITSs (Evens *et al.* 1993; Alevan 2001; Graesser *et al.* 2001) explores NL as one of the keys to bridge the gap between current ITSs and their human counterparts (Anderson *et al.* 1995).

We report on our approach to adding Natural Language Generation (NLG) capabilities to an existing ITS. Our choice has been to apply simple NLG techniques to improve the feedback provided by the ITS. This approach is due to our desire to assess how effective the system can be using inexpensive NLG techniques. This specific project is part of a larger research program whose goals include uncovering the characteristics of tutoring dialogues which foster learning the most and modeling them in NL interfaces to ITSs (Di Eugenio 2001).

We have built two versions of the system, DIAG-NLP1 and DIAG-NLP2, that must present aggregate content. They are both built on top of the EXEMPLARS sentence planner by CogenTex (White & Caldwell 1998). We have conducted a formal evaluation of DIAG-NLP1 by pitting it against the original system and have shown that DIAG-NLP1 on the whole provides better instructional feedback than the original system. We describe DIAG-NLP1 and that evaluation in (Di Eugenio, Glass, & Trolino 2002). In this paper, we discuss DIAG-NLP2.

We show how relatively inexpensive domain-independent techniques for text structuring and for referential expression

generation can be used to make aggregate content more fluent and comprehensible; and how specific rhetorical relations such as *contrast* and *concession* can arise from the data itself, introduced in a bottom-up fashion rather than being planned top-down by a discourse planner. This is an important advancement in the development of DIAG-NLP2 (anonymous).

In DIAG-NLP2, we generate language by coupling the EXEMPLARS sentence planner with the SNePS Knowledge Representation and Reasoning System (Shapiro 2000). SNePS allows us to recognize structural similarities easily, use shared structures, and refer to whole propositions.

While of course we do not advocate eliminating robust text planning modules from NLG systems, we show that in cases like ours, in which the back-end system provides fairly structured content to be communicated in independent turns, text coherence and fluency can be achieved with relatively inexpensive techniques that work locally.

Motivation

The context of our work are ITSs that teach students to troubleshoot systems such as home heating and circuitry, built via the DIAG shell (Towne 1997). A typical session with a DIAG application presents the student with a series of troubleshooting problems of increasing difficulty. The student tests indicators and tries to infer which faulty part (RU) may cause the detected abnormal states. RU stands for *replaceable unit*, because the only course of action for the student to fix the problem is to replace faulty components in the graphical simulation. At any point, the student can consult the built-in tutor in one of several ways. For example, if an indicator shows an abnormal reading, s/he can ask the tutor for a hint regarding which part may cause the problem.

The DIAG shell has some primitive language generation facilities available: the answer in Figure 1 is a representative example of what DIAG can generate, and motivates the need for aggregation. The output is produced in response to a query about why the oil is not flowing to the oil pump. One simple way to make the text more understandable is to order the sentences according to units that always, sometimes, or never produce the abnormality. However, the redundancy in sentences 2-8 still makes the text difficult to understand.

In DIAG-NLP1, the goal was to aggregate information presented by the tutor. Specifically, we focused on syn-

