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; Aleven 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, & Trolio 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-

```
1  The Oil flow indicator is not flowing
   which is abnormal in startup mode (nor-
   mal is flowing).
2  Oil Nozzle always produces this abnormal-
   ity when it fails.
3  Oil Supply valve always produces this ab-
   normality when it fails.
4  Oil Pump always produces this abnormality
   when it fails.
5  Oil Filter always produces this abnormal-
   ity when it fails.
6  System Control Module sometimes produces
   this abnormality when it fails.
7  Burner Motor always produces this abnor-
   mality when it fails.
8  Ignitor assembly never produces this ab-
   normality when it fails.
```

Figure 1: A response from a DIAG tutor in the home heating system domain

```
1   The Oil flow indicator is not flowing
    which is abnormal in startup mode.
2   Normal in this mode is flowing.
3   Within the Oil Burner
4     These replaceable units always produce
    this abnormal indication when they fail:
5       Oil Nozzle;
6       Oil Supply Valve;
7       Oil pump;
8       Oil Filter;
9       Burner Motor.
10  The Ignitor assembly replaceable unit
    never produces this abnormal indication
    when it fails.
11  Within the Furnace System
12    The System Control Module replaceable
    unit sometimes produces this abnormal in-
    dication when it fails.
```

Figure 2: The same response by DIAG-NLP1

tactic aggregation (Dalianis 1996; Huang & Fiedler 1996; Shaw 1998; Reape & Mellish 1998); what we call *functional aggregation*, namely, grouping parts according to the structure of the system; and improving the format of the output. Figure 2 gives a response generated by DIAG-NLP1.

The response aggregates information about replaceable units first by subsystem of the heating system (in this example oil burner and furnace) and then by the certainty with which the unit, if it has failed, might result in the system indication ("always", "often", "sometimes", "never"). We will call these the *system* and *certainty dimensions* of our aggregation. We will refer to the actual values of system and values of certainty that we aggregate units into as the *system* and *certainty dimension values*. Although the aggregation imposes an organization on the information, it still fails to make that organization quickly understandable and useful for troubleshooting the system. For example, it is easy to overlook the transition between dimensional certainty values "never" and "always" in going from 4-9 to 10. Figure 3 gives

```
1  The Oil Flow indicator is not flowing
   in startup mode.
2  This is abnormal.
3  Normal in this mode is flowing.

4  Within the Furnace System,
       this is sometimes caused if
5      the system control module has failed.

6  Within the oil burner,
       this is never caused if
7      the ignitor assembly has failed.

8  In contrast, this is always caused if
9      the burner motor, oil filter, oil
       pump, oil supply valve or oil nozzle
       has failed.
```

Figure 3: A response from the DIAG tutor in the home heating system domain-DIAG-NLP2

a response generated by DIAG-NLP2 for the same interaction. Note that the aggregate structure is still the same. However, the CONTRAST rhetorical relation (Mann & Thompson 1988) is used between units that "never" (lines 6–7) and units that "always" (lines 8–9) cause the indication.

At first glance, it might seem that the system must formulate a goal to impress the student with the importance of some of these dimensional values. However, DIAG-NLP2 highlights the dimensional structure of the aggregation and their values using relatively inexpensive techniques for text structuring and for referential expression generation, a more robust knowledge representation of the domain, and a small amount of lexical information. We believe no other work on aggregation introduces rhetorical relations that stress the relationship between scalar values as we do here.

Two other changes that we made in our second prototype have improved the system's feedback capabilities as well: we prefer aggregations that have fewer dimensional values as the first dimension to present, and we perform referential expression generation, including references to whole prepositions (*discourse deixis*, cf. (Webber 1991)).

In DIAG-NLP1, information is always aggregated first by subsystem and second by certainty. In DIAG-NLP2, we select the aggregation with the smaller top-level branching factor. (*System* is the default if there is a tie.) The intuition is that the top-level dimension of the aggregation should have as few dimension values as possible so as not to overwhelm the student with value categories. Moreover, when the dimension values are scalar, and there are several items (more than 2) that fall under one dimensional value, it appears to be important to highlight this aggregation with a summary statement.

Figure 4 shows DIAG-NLP1's response using system first and then certainty even though every unit mentioned never has any effect on the state of the water temperature gauge. The breakdown by system first implies that there is a purpose in making this distinction when there is not. In contrast, Figure 5 shows the response of DIAG-NLP2 for the same interaction. Prototype 2 selects the aggregation by certainty

```
1    The Water Temperature Gauge indicator is
     100 which is normal in startup mode.
2    Within the Oil Burner
3      These replaceable units have no effect
     on this indicator:
4        Oil Nozzle;
5        Oil Supply Valve;
6        Oil pump;
7        Oil Filter;
8        Ignitor assembly;
9        Burner Motor.
10   Within the Furnace System
11     The System Control Module replaceable
       unit has no effect on this indicator.
```

Figure 4: Aggregation can give the wrong impression - DIAG-NLP1

```
1   The water temperature gauge indicator is
    100 in startup mode.
2   This is normal.

3   The water temperature gauge indicator at
    100 in startup mode would never be
    affected even if
4   one of the following replaceable units has
    failed:

5       within the Furnace System,
          the system control module;

6       within the oil burner,
          the ignitor assembly, burner motor,
          oil filter oil pump, oil supply
          valve, or oil nozzle.
```

Figure 5: Better aggregation - DIAG-NLP2

first (top-level branching factor of one) and produces a summary statement (lines 3-4) before listing the seven RUs under the scalar dimensional value "never" (lines 5-6). Doing so does not highlight the breakdown by system, even though it is still expressed (lines 5 and 6) and instead, emphasizes a summary point: none of the units effect the indication being questioned.

Moreover, whereas in DIAG-NLP1 referential expressions were generated ad hoc, in DIAG-NLP2 we implemented the GNOME algorithm to generate referential expressions (Kibble & Power 2000), a simple algorithm that uses insights from centering (Grosz, Joshi, & Weinstein 1995) and from theories of salience. Importantly, the SNePS formalism allows us to treat propositions as discourse entities that can be added to the discourse model. The GNOME algorithm is then used to generate references to those propositions, such as *this* in *this is always caused* in line 4, Figure 3; *this* refers to the entire clause expressed in line 1.

To summarize, in DIAG-NLP2 we use simple text structuring in terms of rhetorical relations such as CONTRAST and CONCESSION to highlight distinctions between dimensional values. We also consider alternative aggregations of the content, preferring aggregations that have fewer di-

```
{ , [{ m32!, [{never},
              {always},
              {sometimes,  [{m93!}] }] },
     { m27!, [{never,      [{m96!}] },
              {always,     [{m99!}, {m90!},
                             {m87!}, {m84!},
                             {m81!}] },
              {sometimes}] }] }
```

Figure 6: Aggregation first by system then certainty

mensional values as the first dimension to present. In particular, if a dimension has only has a scalar dimension value that more than two units fall into, presenting it with a summary statement improves reader comprehension of the aggregation. Finally, we perform generation of referential expressions, including references to entire propositions.

Making Distinctions between Dimensional Values with CONTRAST

When the student consults the system, the `Client` builds several propositions like the one in the semantic network and passes the name of the indicator questioned, for example `Oil Flow`, to our text structurer. The initial step is to query the network for a unit (represented by a base node) with the name `Oil Flow` and to establish that it is an indicator. This returns a unique base node. Because all base nodes in the network are unique, all rules about the certainty with which various replaceable units affect a base node can be retrieved in one query. We will call these rule nodes the *certainty rules*.

The certainty rules are the units that we aggregate. We use our own Java class, `Aggregation`, to hold the SNePS nodes after they are aggregated. The structure of an `Aggregation` object is like a decision tree. Each tier is an attribute with division of the units by attribute value.

For example, the aggregation that is used as the core content for the text generated in Figure 3 is shown in Figure 6. The first division is by nodes `m32!` and `m27!`. `m32!` asserts that the RU embedded in certainty rule `!m93` is a component of the system control module, and `m27!` asserts that RUs embedded in the set of certainty rules `m81!`, `m84!`, `m87!`, `m90!`, `m93!`, `m96!`, and `m99!` are all components of the oil burner. Although there are other systems in the overall heating system, only the system control module and the oil burner have component RUs that are referred to in this set of certainty rules. The second division of of the certainty rules is by the scalar values used as certainties in the rules: `never`, `sometimes`, and `always`.

We build a text structure using an aggregation. Because the divisions in an aggregation are arbitrary from a rhetorical point of view, we use JOINT, a multinuclear schema that has no corresponding relation. There is one exception to this rule. When the text structurer encounters an aggregation with precisely two attribute values that are scalar, like "never" and "sometimes", or "always" and "never", etc. is uses the binuclear CONTRAST relation instead. As an example, Figure 7 shows the first stage of what the text struc-

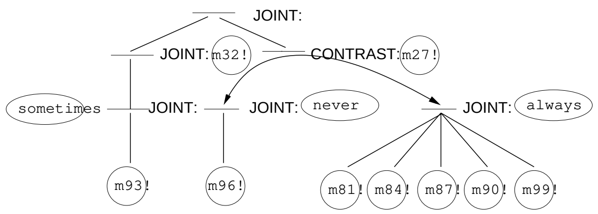

Figure 7: The text structure built from Figure 6, Stage 1

```
1   The oil flow indicator is flowing in
    startup mode.
2   This is normal.

3   The oil flow indicator flowing in startup
    mode would never be affected even if
4   one of the following replaceable units has
    failed:

5     within the living room,
        the IR sensor or room water supply
        valve;

6     within the water pump and safety cutoff
      valve,
        the water temperature safety cutoff
        valve or water pump;

7   In contrast, within the furnace system,
    this would often be affected if
8   the system control module has failed.
```

Figure 8: Another response – DIAG-NLP2

turer builds for the aggregation in Figure 6. We note that the CONTRAST relation is just as compelling when the scalar values that are used are not at extremes, or even equidistant from the center of the scale to which they belong. For example, the CONTRAST relation works well between lines 3–6 and lines 7–8 in the example output in Figure 8 where the contrast is between RUs that “never” have an effect and units that “often” have an effect on the state of indicator.

Minimizing Aggregate Branching at the Top-level

Whenever we construct an aggregation for some content, we construct two: 1) by system and then certainty and 2) by certainty then system. An aggregation by system and then certainty was given in Figure 6. Figure 9 shows the same content aggregated by certainty and then system. Since the top-level branching factor in Figure 9 is 3 the aggregation in Figure 6 (top-level branching factor of 2) is selected.

We note that presenting the aggregation in either order is acceptable in this example. However, when there is an aggregation based on a scalar value where several units fall under one dimension value, it becomes misleading to not present it early. As discussed earlier, the presentation in Figure 4 is confusing because all the units have no effect, yet the top-level presentation of them is by system. When the

```
{ , [{never,     [{m32!},
                   {m27!, [{m96!}] }] },
      {always,    [{m32!},
                   {m27!, [{m99!}, {m90!},
                   {m87!}, {m84!}, {m81!}] }] },
      {sometimes, [{m32!, [{m93!}] },
                    {m27!}] }] }
```

Figure 9: Aggregation by certainty then system

dimension aggregated on has scalar values (“always”, “often”, “sometimes”, “never”) and several fall in under one dimension value (more than two), we modified our text structurer to include a summary statement (Figure 5, line 3 and 4) for the dimension value, followed by a list of units, even if there is further aggregation by system. We believe that the texts become misleading if we do not do this because the semantics of scalar values is powerful forcing highlighting of an aggregation where when several units fall under a scalar value. Figure 8 illustrates that a summary statement and list (lines 3-7) is still comprehensible and effective even though it is embedded in a CONTRAST relation between (3-7) and (8-9).

Related Work

Our approach is similar to that taken by systems such as FOG (Goldberg, Driedger, & Kittredge 1994), that is, systems that receive a fairly structured input from their backend, and where the emphasis is on sentence planning, rather than on text planning to achieve full rhetorical goals such as persuading the hearer to take a certain action. Work by Sibun (1992) is relevant as well. Sibun advocates an approach in which text coherence is parasitic to the subject matter, and in which text is locally organized, without any additional rhetorical structure.

In contrast to both (Goldberg, Driedger, & Kittredge 1994) and (Sibun 1992), however, we do make use of limited text structuring, and introduce few rhetorical relations to make the structure of the aggregate content clear and to highlight the important features of it. Our approach is different from choosing syntactic embedding to perform aggregation as in e.g. (Scott & Sieckenius de Souza 1990), in which the text planner does build a rhetorical representation of the text.

In more recent work, an approach similar to ours— using simple generation techniques for sentence planning — is taken in YAG (McRoy, Channarukul, & Ali 2000). YAG is a system that uses templates like EXEMPLARS, and is intended to be used for tutoring dialogues. However, a problem with templates is that they potentially proliferate. The inheritance mechanism in EXEMPLARS partly prevents this problem and was selected as our text planning formalism for that reason.

Looking more closely at the phenomena our aggregation module deals with, part of it (the *certainty* dimension) concerns standard types of aggregation such as simple conjunction and conjunction via shared participants (Reiter & Dale 2000), as done by (Dalianis 1996; Huang & Fiedler 1996; Shaw 1998). However, what we call *functional aggrega-*

tion (the *system* dimension) introduces semantic elements that are outside the purview of syntactic aggregation: perhaps it can be considered as a type of *conceptual aggregation* (Reape & Mellish 1998). Such functional aggregation appears to be preferred by humans over syntactic aggregation (see (Di Eugenio, Glass, & Trolio 2002)). We believe no other work on aggregation introduces rhetorical relations as we do here, however this appears to be appropriate whenever scalar values are to be aggregated over.

Conclusions and Future Work

We have presented our approach to improving the generation of aggregate content in the context of the feedback produced by an ITS. We have shown how relatively inexpensive domain-independent techniques for text structuring and referential expression generation can be used to make the text that expresses aggregate content more fluent and comprehensible. We have also shown how text structuring is determined bottom up by the specifics of the text to be aggregated, specifically, by the relationships between scalar values.

Although the NLG module faces a simplified task in our case, we contend that our approach is appropriate for systems in which the back-end provides fairly structured content to be communicated in independent turns. In particular, whenever scalar values need to be aggregated, it is possible to introduce rhetorical relations that stress the relationship between those values.

We are currently running a user study to evaluate DIAG-NLP2, as we did for DIAG-NLP1 (Di Eugenio, Glass, & Trolio 2002). Various metrics are collected, both objective such as time on task and subjective such as rating the system's feedback on a scale from 1 to 5. We will compare the results obtained with DIAG-NLP2 with those obtained for DIAG-NLP1 and for the original system, which is the baseline.

References

Aleven, V., ed. 2001. San Antonio, TX: The International Society of Artificial Intelligence in Education.

Anderson, J. R.; Corbett, A. T.; Koedinger, K. R.; and Pelletier, R. 1995. Cognitive tutors: Lessons learned. *Journal of the Learning Sciences* 4(2):167–207.

Dalianis, H. 1996. *Concise Natural Language Generation from Formal Specifications*. Ph.D. Dissertation, Department of Computer and Systems Science, Stocholm UNiversity. Technical Report 96-008.

Di Eugenio, B.; Glass, M.; and Trolio, M. 2002. The diag experiments: Natural language generation for intelligent tutoring systems. In *INLG02, The Second International Natural Language Generation Conference*, 120–127.

Di Eugenio, B. 2001. Natural language processing for computer-supported instruction. *Intelligence* 12(4).

Evens, M. W.; Spitkovsky, J.; Boyle, P.; Michael, J. A.; and Rovick, A. A. 1993. Synthesizing tutorial dialogues. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, 137–140. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Goldberg, E.; Driedger, N.; and Kittredge, R. 1994. Using natural language processing to produce weather forecast. *IEEE Expert* 9:45–53.

Graesser, A.; VanLehn, K.; Rosé, C. P.; Jordan, P. W.; and Harter, D. 2001. Intelligent tutoring systems with conversational dialogue. *AI Magazine* 22(4):39–52.

Grosz, B.; Joshi, A.; and Weinstein, S. 1995. Centering: A Framework for Modeling the Local Coherence of Discourse. *Computational Linguistics* 21(2):203–225.

Huang, X., and Fiedler, A. 1996. Paraphrasing and aggregating argumentative text using text structure. In *Proceedings of the 8th Int. Workshop on NLG*, 21–30.

Kibble, R., and Power, R. 2000. Nominal generation in GNOME and ICONOCLAST. Technical report, Information Technology Research Institute, University of Brighton, Brighton, UK.

Mann, W. C., and Thompson, S. A. 1988. Rhetorical structure theory: Towards a functional theory of text organization. *TEXT* 8(3):243–281.

McRoy, S. W.; Channarukul, S.; and Ali, S. 2000. Text realization for dialog. In *Dialogue Systems for Tutorial Applications*. AAAI Fall Symposium.

Reape, M., and Mellish, C. 1998. Just what *is* aggregation anyway? In *Proceedings of the European Workshop on Natural Language Generation*.

Reiter, E., and Dale, R. 2000. *Building Natural Language Generation Systems*. Studies in Natural Language Processing. Cambridge University Press.

Scott, D., and Sieckenius de Souza, C. 1990. Getting the message across in RST-based text generation. In Dale, R.; Mellish, C.; and Zock, M., eds., *Current Research in Natural Language Generation*. Academic Press.

Shapiro, S. C. 2000. SNePS: A logic for natural language understanding and commonsense reasoning. In Iwanska, L. M., and Shapiro, S. C., eds., *Natural Language Processing and Knowledge Representation*. AAAI Press/MIT Press.

Shaw, J. 1998. Segregatory coordination and ellipsis in text generation. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics*, 1220–1226.

Sibun, P. 1992. Generating Text without Trees. *Computational Intelligence: Special Issue on Natural Language Generation* 8(1).

Towne, D. M. 1997. Approximate reasoning techniques for intelligent diagnostic instruction. *International Journal of Artificial Intelligence in Education*.

Webber, B. L. 1991. Structure and Ostension in the Interpretation of Discourse Deixis. *Language and Cognitive Processes* 6(2):107–135.

White, M., and Caldwell, T. 1998. Exemplars: A practical, extensible framework for dynamic text generation. In *Proceedings of the Ninth International Workshop on Natural Language Generation*.