# The DIAG experiments: Natural Language Generation for Intelligent Tutoring Systems

**Barbara Di Eugenio** and **Michael Glass** and **Michael J. Trolio**
Computer Science Department
University of Illinois
Chicago, IL, 60607, USA
{bdieugen,mglass}@cs.uic.edu

**Abstract**

We added a sentence planning component to an existing ITS that teaches students how to troubleshoot mechanical systems. We evaluated the original version of the system and the enhanced one via a user study in which we collected performance, learning and usability metrics. We show that on the whole the enhanced system is better than the original one. We discuss how to use the binomial cumulative distribution to assess cumulative effects.

## 1 Introduction

Intelligent Tutoring Systems (ITSs) help students master a certain topic. Research on the next generation of ITSs (Evens et al., 1993; Rosé and Freedman, 2000; Aleven, 2001; Graesser et al., 2001) explores NL as one of the keys to bridge the gap between current ITSs and human tutors (Anderson et al., 1995).

Our work is the first to show that it is the NL interaction that improves students' learning or at least the students' experience with the system. We added NLG capabilities to an existing ITS. We focused on sentence planning, and specifically, on aggregation. We then conducted a systematic evaluation that pitted the original version of the system against the enhanced one. We show that on the whole the enhanced system outperforms the original one.

Our work is also relevant to evaluating NLG systems in general, as we show how to use the binomial cumulative distribution function to assess cumulative effects.

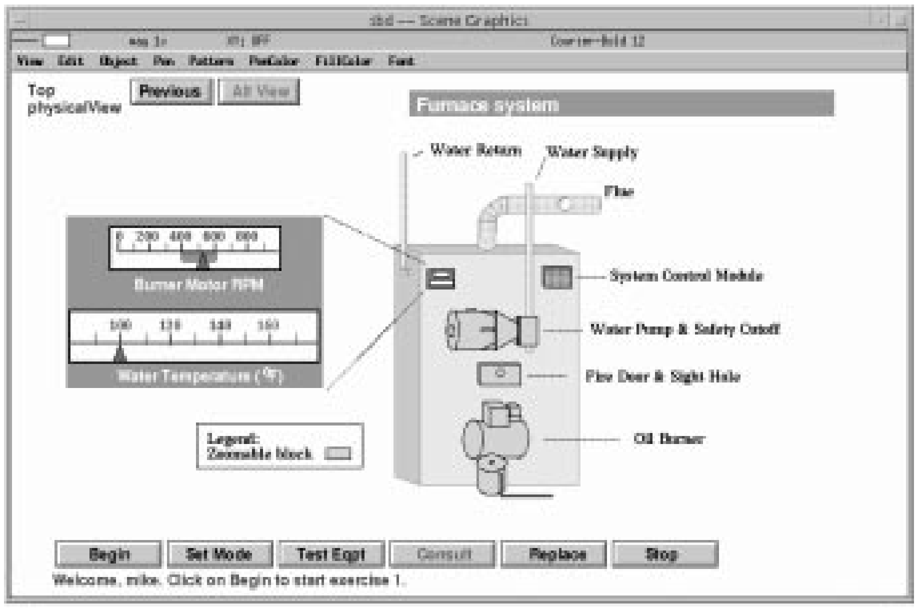
We will first discuss DIAG, the ITS we are using, and the sentence planning component we added to DIAG. We will then describe the formal evaluation we conducted. We will conclude by discussing related work and our current work.

## 2 Language Generation for DIAG

DIAG (Towne, 1997) is a shell to build ITSs that teach students to troubleshoot complex systems such as home heating and circuitry. Authors build interactive graphical models of systems, and build lessons based on these graphical models (see Figure 1).

A DIAG application presents a 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. Figure 1 shows the furnace system, one subsystem of the home heating system in our DIAG application. Figure 1 includes indicators such as the gauge labeled Water Temperature, replaceable units, and other complex modules (Oil Burner) that contain indicators and replaceable units. Complex components are zoomable.

At any point, the student can consult the built-in tutor via the Consult menu (cf. the Consult button in Figure 1). For example, if an indicator shows an abnormal reading, s/he can ask the tutor for a hint regarding which RUs may cause the problem. After deciding which content to communicate, the original DIAG system (*DIAG-orig*) uses very simple templates to assemble the text to present to the student.



Welcome, mike. Click on Begin to start exercise 1.

Figure 1: A screen from a DIAG application on home heating

As a result, the feedback provided by *DIAG-orig* is repetitive, both inter- and intra-turn. In many cases, the feedback presents a long list of parts. The top part of Figure 2 shows the reply provided by *DIAG-orig* to a request of information regarding the indicator named “Visual Combustion Check”.

### 2.1 The sentence planner

We set out to rapidly improve DIAG’s feedback mechanism. Our main goals were to to assess whether simple NLG techniques would lead to measurable improvements in the system’s output, and to conduct a systematic evaluation that would focus on language only. Thus, we did not change the tutoring strategy, or alter the interaction between student and system in any way. Rather, we concentrated on improving each turn by avoiding excessive repetitions. We chose to achieve this by: introducing syntactic aggregation (Dalianis, 1996; Huang and Fiedler, 1996; Shaw, 1998; Reape and Mellish, 1998) and what we call *functional aggregation*, namely, grouping parts according to the structure of the system; and improving the format of the output. The bottom part of Figure 2 shows the revised output produced by *DIAG-NLP*. The RUs under discussion are grouped by the system modules that contain them (Oil Burner and Furnace System), and by the likelihood that a certain RU causes the observed symptoms. In contrast to the original answer, the revised answer singles out the *Ignitor Assembly*, the only RU that cannot cause the symptom.

As our sentence planner, we use EXEMPLARS (White and Caldwell, 1998), an object-oriented, rule based generator. It mixes template-style and more sophisticated types of text planning. The rules (called *exemplars*) are meant to capture an exemplary way of achieving a communicative goal in a given communicative context. The text planner selects rules by traversing the exemplar specialization hierarchy, and evaluating the applicability conditions associated with each exemplar.

In *DIAG-NLP*, exemplars are of two main types, description and aggregation / layout. The four description exemplars are used when the full description of a part is required, such as whether the part is in a normal state, its current reading, and, if abnormal, what the normal state should be. The eight aggregation exemplars are used to group large lists of parts into smaller lists. They allow composite aggregation, so that nested lists are created. The topmost aggregation exemplar is *AggByType*, which controls nesting of embedded lists. *AggByType* has 6 daughters, among them: *AggByType* that groups parts by part type, i.e., separates indicators from RUs; *AggByContainer* that accepts a list of parts, classifies

```
The visual combustion check is igniting which is abnormal in this startup mode (normal
is combusting)
Oil Nozzle always
  produces this abnormality when it fails.
Oil Supply Valve always
  produces this abnormality when it fails.
Oil pump always
  produces this abnormality when it fails.
Oil Filter always
  produces this abnormality when it fails.
System Control Module sometimes
  produces this abnormality when it fails.
Ignitor Assembly never
  produces this abnormality when it fails.
Burner Motor always
  produces this abnormality when it fails.
and, maybe others affect this test.
```

```
The visual combustion check indicator is igniting which is abnormal in startup mode.
Normal in this mode is combusting.

Within the Oil Burner
   These replaceable units always produce this abnormal indication when they fail:
     Oil Nozzle;
     Oil Supply Valve;
     Oil pump;
     Oil Filter;
     Burner Motor.

     The Ignitor assembly replaceable unit never produces this abnormal indication
when it fails.

Within the furnace system,
  The System Control Module replaceable unit sometimes produces this abnormal
   indication when it fails.

Also, other parts may affect this indicator.
```

Figure 2: Original (top) and enhanced (bottom) answers to the same *Consult Indicator* query

each part by the system module that contains it, and then creates a set of lists by module; *AggByFufer* that groups RUs according to the likelihood of being at fault for a specific symptom. The eighth aggregation exemplar deals with formatting, namely, creating vertical lists, spacing, etc.

The most frequent application of the aggregation rules is to group parts according to the system module they belong to, and within each module, to group RUs by how likely it is they may cause the observed symptom, cf. Figure 2.

Figure 3 illustrates a simple exemplar for describing one indicator within a list of items.[1] An exemplar is a Java class with an `evalConstraints()` method as its applicability condition, and an `apply()` method, responsible for generating an XML representation of the desired text. Portions delimited by `<<+` and `+>>` are annotated XML statements; braces invoke subsidiary exemplars for generating included text.

The implementation took a graduate student six months. Most of the effort was devoted not to writing exemplars, but to making DIAG and EXEMPLARS communicate.

After a student query, DIAG collects all the information it needs to communicate to the student, and writes it to a text file that is then passed to EXEMPLARS. A portion of the text file that DIAG passes to EXEMPLARS for the example in Figure 2 is as follows (`ConsultIndicator` is the type of query asked by the student):

[1]An exemplar that performs aggregation would be more relevant. However, for expository purposes and because of space constraints we chose a much simpler description exemplar.

```
exemplar DescribeIndicator(Vector lists, int index, String tense)
   extends DescribePart
{
 boolean evalConstraints() {
  return ((Part)lists.elementAt(index) instanceof Indicator);}

 void apply() {
  Indicator ind = (Indicator)lists.elementAt(index);
  <<+The {ind.getName()} indicator {tense} {ind.getState()}+>>
  if ((ind.getState()).equals(ind.getNormalState()))
    <<+ which is normal in {ind.getMode()} mode.+>>
  else
    <<+ which is abnormal in {ind.getMode()} mode.^Normal in this
             mode is {ind.getNormalState()}.+>>}}
```

Figure 3: Exemplar for describing an indicator

```
ConsultIndicator Indicator
name Visual combustion check
state igniting
modeName startup
normalState combusting
-- --
ConsultIndicator ReplUnit
name Oil Nozzle
fufer always
-- --
ConsultIndicator ReplUnit
name System Control Module
fufer sometimes
-- --
ConsultIndicator ReplUnit
name Ignitor assembly
fufer no effect
```

The attribute `fufer`[2] represents the strength of the causal connection between the failure of an RU and an observed symptom.[3] The order of the information in the text file mirrors the order in which DIAG assembles the information, which is also directly mirrored in the feedback provided by *DIAG-orig* (see top of Figure 2).

EXEMPLARS performs essentially three tasks: 1) it determines the specific exemplars needed; 2) it adds the chosen exemplars to the sentence planner as a goal; 3) it linearizes and lexicalizes the feedback in its final form, writing it to a file which is passed back to DIAG for display in the appropriate window.

In this version of *DIAG-NLP*, morphology, lexical realization and referring expression generation were all directly encoded in the appropriate exemplars. We already have a third version of the system in which referring expressions are generated in a principled way, see Section 5.

## 3 Evaluation

Our empirical evaluation is a between-subject study: one group interacts with *DIAG-orig* and the other with *DIAG-NLP*. The 34 subjects (17 per group) were all science or engineering majors affiliated with our university. Each subject read some short material about home heating, went through the first problem as a trial run, then continued through the curriculum on his/her own. The curriculum consists of three problems of increasing difficulty. As there was no time limit, every student solved every problem. At the end of the experiment, each subject was administered a questionnaire.

A log was collected for each subject including, for each problem: whether the problem was solved; total time, and time spent reading feedback; how many and which indicators and RUs the subject consults DIAG about; how many, and which RUs the subject replaces.

The questionnaire is divided into three parts. The first part tests the subject's understanding of the domain. Because the questions are open ended, this part was scored as if grading an essay. The second part asks the subject to rate the system's feedback along four dimensions on a scale from 1 to 5 (see Table 3). The third part concerns whether subjects remember their actions, specifically, the RUs they replaced. We quantify the subjects' recollections in terms of precision and recall with respect to the log

[2]The name comes from DIAG.

[3]These strenghts are entered at development time via the specialized editors that the DIAG authoring system provides.

that the system collects. In Table 2, we report the F-measure ($\frac{(\beta^2+1)PR}{\beta^2P+R}$, with $\beta = 1$) that smooths precision and recall.

## 3.1 Results

Tables 1, 2, and 3 show the results for the cumulative measures across the three problems (individual problems show the same trends).

| | DIAG-orig | DIAG-NLP |
|---|---|---|
| Total Time | 29.8’ | 28.0’ |
| Feedback Time | 6.9’ | 5.4’ |
| Indicator consultations | 11.4 | 5.9 |
| RU consultations | 19.2 | 18.1 |
| Parts replaced | 3.85 | 3.33 |

Table 1: Performance measures

| | DIAG-orig | DIAG-NLP |
|---|---|---|
| Essay score | 81/100 | 83/100 |
| RU recollection | .72 | .63 |

Table 2: Learning and recollection measures

| | DIAG-orig | DIAG-NLP |
|---|---|---|
| Usefulness | 4.35 | 4.47 |
| Helped stay on right track | 4.35 | 4.35 |
| Not misleading | 4.00 | 4.12 |
| Conciseness | 3.47 | 3.76 |

Table 3: Usability measures

Although individually all but one or two measures favor *DIAG-NLP*, differences are not statistically significant. *Indicator consultations* comes closest to significance with a non-significant trend in favor of *DIAG-NLP* (Mann-Whitney test, *U=98, p=0.11*).

We therefore apply the binomial cumulative distribution function (BCDF) —the one-tailed Sign Test (Siegel and Castellan, 1988) —to assess whether *DIAG-NLP* is better than *DIAG-orig*. This test measures the likelihood that *DIAG-NLP* could have beat *DIAG-orig* on $m$ or more out of $n$ independent measures under the null hypothesis that the two systems are equal. This test is insensitive to the magnitude of differences in each measure, noticing only which condition represents a win ((Di Eugenio et al., 2002) discusses the BCDF further).

Table 4 combines the independent measures from Tables 1, 2, and 3, showing which condition was

| | DIAG-orig | DIAG-NLP |
|---|---|---|
| Total Time | | $\surd$ |
| Indicator consultations | | $\surd$ |
| RU consultations | | $\surd$ |
| Parts replaced | | $\surd$ |
| Essay score | | $\surd$ |
| RU recollection | $\surd$ | |
| Usefulness | | $\surd$ |
| Helped stay on right track | $\surd$ | $\surd$ |
| Not misleading | | $\surd$ |
| Conciseness | | $\surd$ |

Table 4: Successes for each system

more successful. The result shows 9/10 (or 8/10) wins for *DIAG-NLP*. Since one measure was tied, we report two sets of probabilities assuming that the tied measure favored *DIAG-orig* or *DIAG-NLP* respectively.

The probability of 9/10 (or 8/10) successes for *DIAG-NLP* under the null hypothesis is $p = 0.01$ (or $0.054$), showing a significant (or marginally significant) win for *DIAG-NLP*. If we question whether *Total Time* is independent of the other measures, then $p = 0.02$ (or $0.09$) for 8/9 (or 7/9) wins, which is at best a statistically significant and at worst a marginally significant win for *DIAG-NLP*.

Had we followed the customary practice of discarding the tied measure (Siegel and Castellan, 1988),[4] *DIAG-NLP* would win 8/9, $p = 0.02$, or 7/8, $p = 0.035$ (depending on the inclusion of *Total Time*), which are both significant.

We can then conclude that the better measures for *DIAG-NLP*, albeit individually not statistically significant, cumulatively show that *DIAG-NLP* outperforms *DIAG-orig*.

## 3.2 Discussion

Our work contributes to both evaluating NLG for ITSs, and evaluating NLG in general.

We developed the set of metrics we collected partly based on the literature (e.g., time on task), partly because of their significance for troubleshooting (e.g., parts replaced). *RU recollection*, which measures what students remember of their own actions, was suggested to us by a colleague in psychology as a possible correlate of learning. Among the

[4]Discarding tied measures appears to be discarding support for the null hypothesis, so we do not argue for this approach (Di Eugenio et al., 2002).

measures we collected, only *Essay score* (and possibly *RU recollection*) directly addresses learning, the others pertain to task performance or user satisfaction. ITSs are frequently evaluated in terms of pre-/post-test scores, where the same test is given to the student before and after using the ITS. In our case, the most appropriate pre-/post-test would have been a troubleshooting problem. However, we felt the binary measure "problem solved/ not solved" would be too rough an assessment to be useful, not to mention that this choice would have meant one fewer problem in the curriculum.

We contend that performance and usability measures are important for an ITS, as they provide indirect evidence of its effectiveness. For example, the lower number of indicator and RU consultations in *DIAG-NLP* is evidence in favor of the effectiveness of the aggregated feedback: because the feedback highlights what is important, subjects can focus their troubleshooting without asking as many questions of the system. Equally important is the lower number of RU replacements. This metric includes the mistakes a student makes, i.e., the parts replaced that are not responsible for the problem. When repairing a real system, replacing parts that are actually working should clearly be kept to a minimum. We claim that an ITS whose NL feedback leads the student more effectively towards the solution of a problem is a better ITS, even if students learn as much in either version of the ITS. This holds for usability as well. In a real setting, students should be more willing to sit down with a system that they perceive as more friendly and usable than a system that engenders similar learning gains, but is harder to use.

Concerning evaluation of NLG in general, a common way of assessing whether system B is better than system A is to collect a number of measures, hoping that there will be at least one statistically significant measure in favor of system B and no significant measure in favor of system A. However, reality is often murkier than this ideal result. A typical result of an evaluation may be that out of twelve measures ten favor B and two favor A, but only two show statistical significance and those two point to opposite conclusions. The BCDF is an appropriate way of assessing whether B outperforms A *on the whole*. Using the BCDF addresses a different type of *cumulative effect* than e.g. PARADISE (Walker et al., 1997). This comprehensive framework for dialogue evaluation combines various measures to yield a cumulative score for each of the systems being evaluated. However, the cumulative scores are then arranged in pairs, and their difference tested for statistical significance. The BCDF is used not to obtain a single score, but to assess what the measures collectively say on the performance of the system.

## 4 Related work

Our work touches on three issues: aggregation; evaluation of NLG systems; and work on evaluating NL interfaces for ITSs.

Part of our rules implement standard types of aggregation such as simple conjunction and conjunction via shared participants (Reiter and Dale, 2000). We also introduced what we call *functional aggregation* (perhaps a type of *conceptual aggregation* (Reape and Mellish, 1998)). Although it introduces semantic elements that are outside the purview of syntactic aggregation, it appears to be preferred by humans over syntactic aggregation (see Section 5).

Evaluation is of great interest for the language generation community (Dale and Mellish, 1998), and much progress has been made in the last few years. Language generation systems have been evaluated e.g. by using human judges to assess the quality of the texts produced (Coch, 1996; Lester and Porter, 1997; Harvey and Carberry, 1998); by comparing the system's performance to that of humans (Yeh and Mellish, 1997); or through task efficacy measures (Young, 1997; Carenini and Moore, 2000; Reiter et al., 2001). We have shown how different measures can be combined to assess what they collectively say on the performance of a system.

Regarding evaluation of NL interfaces for ITSs, no experiment like ours has been published that compares two versions of the same system, one of which uses a NL interface.[5] For example, the CIRCSIM-Tutor system (Evens et al., 1993) which teaches medical physiology has been used with medical students, but it was never evaluated vs. a less-sophisticated version; the ANDES system which teaches physics, and its NL version ATLAS (VanLehn et al., 2000) have been evaluated, but only in a

[5]One relevant experiment is (Trafton et al., 1997), but their system does not really qualify as an ITS.

very small pilot study (Graesser et al., 2001).

## 5 Current and future work

We are currently pursuing two lines of research.

First, we added some sophistication to our sentence planner. *DIAG-NLP2* is a third fully implemented version of the system (Haller et al., 2002). *DIAG-NLP2* incorporates the GNOME algorithm (Kibble and Power, 2000) to generate referring expressions, including references to whole propositions such as *This is caused ...*, and models a few rhetorical relations such as *contrast* and *concession*. *DIAG-NLP2* couples EXEMPLARS to a knowledge base built via the SNePS representation system (Shapiro and Rapaport, 1992). SNePS makes it easy to represent and reason about entire propositions, not just about objects.

Second, we have conducted a constrained data collection to uncover empirical evidence for the rules we implemented in EXEMPLARS. Doing the implementation first and then looking for empirical evidence may appear backwards. As one of our goals was to rapidly improve *DIAG-orig*'s output and evaluate the improvement, we could not wait for the result of an empirical investigation. In this, our work follows much work on aggregation (Dalianis, 1996; Huang and Fiedler, 1996; Shaw, 1998), in which aggregation rules and heuristics are plausible, but are not based on any hard evidence. Even when corpus studies are used (Dalianis, 1996; Harvey and Carberry, 1998), they are not completely convincing. Aggregation rules could be posited from a corpus only if we knew the underlying representation the text had been aggregated from, which is usually not the case. The data collection we conducted was meant to address this last issue as well.

To understand how a human tutor may verbalize a collection of facts, we collected 23 tutoring dialogues (for a total of 270 tutor turns) between a student interacting with the DIAG application on home heating and a human tutor. The tutor and the student are in different rooms, sharing images of the same DIAG tutoring screen. When the student consults DIAG, the tutor sees the information that DIAG would use in generating its advice — exactly the same information that DIAG gives to EXEMPLARS in *DIAG-NLP*. The tutor then types a response that substitutes for DIAG's response. Although we cannot constrain the tutor to mention all and only the facts that DIAG would have communicated, we can still analyze how the tutor uses the information provided by DIAG.

We have recently developed a coding scheme and started annotating the data. As a preliminary observation, the most striking pattern is that the humans eschew syntactic aggregation of part lists and instead describe functional aggregations of parts. This lends support to our rule that groups parts according to the system hierarchical structure. For example, the same assemblage of parts, i.e., oil nozzle, supply valve, pump, filter, etc, can be described as *the other items on the fuel line* or as *the path of the oil flow*.

**Acknowledgements.** This work is supported by grants N00014-99-1-0930 and N00014-00-1-0640 from the Office of Naval Research, and partly by grant EIA-9802090 from the National Science Foundation. We are grateful to CoGenTex Inc., in particular to Mike White, for making EXEMPLARS available to us; and to Michael Scott for suggesting the BCDF.

## References

Vincent Aleven, editor. 2001. *Workshop on Tutorial Dialogue Systems*, San Antonio, TX, May. The International Society of Artificial Intelligence in Education.

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

Giuseppe Carenini and Johanna D. Moore. 2000. An empirical study of the influence of argument conciseness on argument effectiveness. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, Hong Kong.

José Coch. 1996. Evaluating and comparing three text-production techniques. In *COLING96, Proceedings of the Sixteenth International Conference on Computational Linguistics*, pages 249–254, Copenhagen, Denmark, August.

Robert Dale and Chris Mellish. 1998. Towards the evaluation of natural language generation. In *Proceedings of the First International Conference on Language Resources and Evaluation*, Grenada, Spain, May.

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

Barbara Di Eugenio, Michael Glass, and Michael J. Scott. 2002. The binomial cumulative distribution, or, is my system better than yours? In *LREC2002, Proceedings of the Third International Conference on Language Resources and Evaluation,* Las Palmas, Spain.

Martha W. Evens, John Spitkovsky, Patrick Boyle, Joel A. Michael, and Allen A. Rovick. 1993. Synthesizing tutorial dialogues. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, pages 137–140, Hillsdale, New Jersey. Lawrence Erlbaum Associates.

Arthur Graesser, Kurt VanLehn, Carolyn P. Rosé, Pamela W. Jordan, and Derek Harter. 2001. Intelligent tutoring systems with conversational dialogue. *AI Magazine*, 22(4):39–52.

Susan Haller, Barbara Di Eugenio, and Michael J. Trolio. 2002. Generating natural language aggregations using a propositional representation of sets. In *FLAIRS 2002, the 15th International Florida AI Research Symposium*, Pensacola Beach, FL, May.

Terrence Harvey and Sandra Carberry. 1998. Integrating text plans for conciseness and coherence. In *ACL/COLING 98, Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics*, pages 512–518, Montreal, Canada.

Xiaoron Huang and Armin Fiedler. 1996. Paraphrasing and aggregating argumentative text using text structure. In *Proceedings of the 8th International Workshop on Natural Language Generation*, pages 21–30, Sussex, UK.

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

James C. Lester and Bruce W. Porter. 1997. Developing and empirically evaluating robust explanation generators: the KNIGHT experiments. *Computational Linguistics*, 23(1):65–102. Special Issue on Empirical Studies in Discourse.

Mike Reape and Chris Mellish. 1998. Just what *is* aggregation anyway? In *Proceedings of the European Workshop on Natural Language Generation*, Toulouse, France.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Studies in Natural Language Processing. Cambridge University Press.

Ehud Reiter, Roma Robertson, A. Scott Lennox, and Liesl Osman. 2001. Using a Randomised Controlled Clinical Trial to Evaluate an NLG System. In *ACL-2001, Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 434–441, Toulouse, France.

Carolyn P. Rosé and Reva Freedman, editors. 2000. *Building Dialogue Systems for Tutorial Applications (AAAI Fall Symposium)*. American Association for Artificial Intelligence.

Stuart Shapiro and William Rapaport. 1992. The SNePS Family. *Computers and Mathematics with Applications, Special Issue on Semantic Networks in Artificial Intelligence, Part 1*, 23(2–5).

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

Sidney Siegel and N. John Castellan, Jr. 1988. *Nonparametric statistics for the behavioral sciences*. McGraw Hill.

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

J. G. Trafton, K. Wauchope, P. Raymond, B. Deubner, J. Stroup, and E. Marsch. 1997. How natural is natural language for intelligent tutoring systems? In *Proceedings of the Annual Conference of the Cognitive Science Society*.

K. VanLehn, R. Freedman, P. W. Jordan, C. Murray, C. Oran, M. Ringenberg, C. P. Rosé, K. Schultze, R. Shelby, D. Treacy, A. Weinstein, and M. Wintersgill. 2000. Fading and deepening: The next steps for ANDES and other model-tracing tutors. In *Proceedings of the Intelligent Tutoring Systems Conference*.

Marilyn A. Walker, Diane J. Litman, Candace A. Kamm, and Alicia Abella. 1997. PARADISE: A Framework for Evaluating Spoken Dialogue Agents. In *ACL-EACL97, Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 271–280.

Michael White and Ted Caldwell. 1998. Exemplars: A practical, extensible framework for dynamic text generation. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, pages 266–275, Niagara-on-the-Lake, Canada.

Ching-Long Yeh and Chris Mellish. 1997. An empirical study on the generation of anaphora in Chinese. *Computational Linguistics*, 23(1):169–190. Special Issue on Empirical Studies in Discourse.

R. Michael Young. 1997. *Generating Descriptions of Complex Activities*. Ph.D. thesis, Intelligent Systems Program, University of Pittsburgh.