Aggregation improves learning: experiments in natural language generation for intelligent tutoring systems

Barbara Di Eugenio and **Davide Fossati** and **Dan Yu**
University of Illinois
Chicago, IL, 60607, USA
{bdieugen,dfossa1,dyu6}@uic.edu

Susan Haller
University of Wisconsin - Parkside
Kenosha, WI 53141, USA
haller@cs.uic.edu

Michael Glass
Valparaiso University
Valparaiso, IN, 46383, USA
Michael.Glass@valpo.edu

Abstract

To improve the interaction between students and an intelligent tutoring system, we developed two Natural Language generators, that we systematically evaluated in a three way comparison that included the original system as well. We found that the generator which intuitively produces the best language does engender the most learning. Specifically, it appears that *functional* aggregation is responsible for the improvement.

1 Introduction

The work we present in this paper addresses three issues: evaluation of Natural Language Generation (NLG) systems, the place of aggregation in NLG, and NL interfaces for Intelligent Tutoring Systems.

NLG systems have been evaluated in various ways, such as via task efficacy measures, i.e., measuring how well the users of the system perform on the task at hand (Young, 1999; Carenini and Moore, 2000; Reiter et al., 2003). We also employed task efficacy, as we evaluated the learning that occurs in students interacting with an Intelligent Tutoring System (ITS) enhanced with NLG capabilities. We focused on sentence planning, and specifically, on aggregation. We developed two different feedback generation engines, that we systematically evaluated in a three way comparison that included the original system as well. Our work is novel for NLG evaluation in that we focus on one specific component of the NLG process, aggregation. Aggregation pertains to combining two or more of the messages to be communicated into one sentence (Reiter and Dale, 2000). Whereas it is considered an essential task of an NLG system, its specific contributions to the effectiveness of the text that is eventually produced have rarely been assessed (Harvey and Carberry, 1998). We found that syntactic aggregation does not improve learning, but that what we call *functional* aggregation does. Further, we ran a controlled data collection in order to provide a more solid empirical base for aggregation rules than what is normally found in the literature, e.g. (Dalianis, 1996; Shaw, 2002).

As regards NL interfaces for ITSs, research on the next generation of ITSs (Evens et al., 1993; Litman et al., 2004; Graesser et al., 2005) explores NL as one of the keys to bridging the gap between current ITSs and human tutors. However, it is still not known whether the NL interaction between students and an ITS does in fact improve learning. We are among the first to show that this is the case.

We will first discuss DIAG, the ITS shell we are using, and the two feedback generators that we developed, *DIAG-NLP1* and *DIAG-NLP2*. Since the latter is based on a corpus study, we will briefly describe that as well. We will then discuss the formal evaluation we conducted and our results.

2 Natural Language Generation for DIAG

DIAG (Towne, 1997) is a shell to build ITSs based on interactive graphical models that teach students to troubleshoot complex systems such as home heating and circuitry. 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 abnormal states detected via the indicator readings. 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.

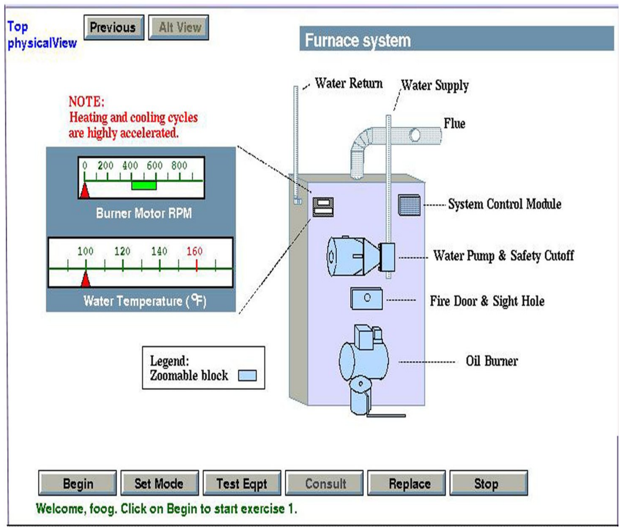

Figure 1: The furnace system

Fig. 1 shows the furnace, one subsystem of the home heating system in our DIAG application. Fig. 1 includes indicators such as the gauge labeled Water Temperature, RUs, and complex modules (e.g., the Oil Burner) that contain indicators and RUs. Complex components are zoomable.

At any point, the student can consult the tutor via the Consult menu (cf. the Consult button in Fig. 1). There are two main types of queries: *ConsultInd(icator)* and *ConsultRU*. *ConsultInd* queries are used mainly when an indicator shows an abnormal reading, to obtain a hint regarding which RUs may cause the problem. DIAG discusses the RUs that should be most suspected given the symptoms the student has already observed. *ConsultRU* queries are mainly used to obtain feedback on the diagnosis that a certain RU is faulty. DIAG responds with an assessment of that diagnosis and provides evidence for it in terms of the symptoms that have been observed relative to that RU.

The original DIAG system (*DIAG-orig*) uses very simple templates to assemble the text to present to the student. The top parts of Figs. 2 and 3 show the replies provided by *DIAG-orig* to a *ConsultInd* on the *Visual Combustion Check*, and to a *ConsultRu* on the *Water Pump*.

The highly repetitive feedback by *DIAG-orig* screams for improvements based on aggregation techniques. Our goal in developing *DIAG-NLP1* and *DIAG-NLP2* was to assess whether simple, rapidly deployable NLG techniques would lead to measurable improvements in the student's learning. Thus, in both cases it is still DIAG that performs content determination, and provides to *DIAG-NLP1* and *DIAG-NLP2* a file in which the facts to be communicated are written – a *fact* is the basic unit of information that underlies each of the clauses in a reply by *DIAG-orig*. The only way we altered the interaction between student and system is the actual language that is presented in the output window. In *DIAG-NLP1* we mostly explored using syntactic aggregation to improve the feedback, whereas *DIAG-NLP2* is corpus-based and focuses on functional aggregation. In both *DIAG-NLP1* and *DIAG-NLP2*, we use EXEMPLARS (White and Caldwell, 1998), an object-oriented, rule-based generator. The rules (called *exemplars*) are meant to capture an exemplary way of achieving a communicative goal in a given context. EXEMPLARS selects rules by traversing the exemplar specialization hierarchy and evaluating the applicability conditions associated with each exemplar.

The visual combustion check is igniting which is abnormal (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.

The visual combustion check indicator is igniting.
This is abnormal.
Normal is combusting.

Within the furnace system,
    this is sometimes caused if
        the System Control Module has failed.

Within the Oil Burner
    this is never caused if
        the Ignitor Assembly has failed.
    In contrast, this is always caused if
        the Burner Motor, Oil Filter, Oil Pump,
        Oil Supply Valve, or Oil Nozzle has failed.

The combustion is abnormal.
In the oil burner, check the units along the path of the oil and the burner motor.

Figure 2: Answers to *ConsultInd* by *DIAG-orig*, *DIAG-NLP1* and *DIAG-NLP2*

Water pump is a very poor suspect.
Some symptoms you have seen conflict with that theory.
Water pump sound was normal.
This normal indication never results when this unit fails.
Visual combustion check was igniting.
This abnormal indication never results when this unit fails.
Burner Motor RMP Gauge was 525.
This normal indication never results when this unit fails.

The Water pump is a very poor suspect.
Some symptoms you have seen conflict with that theory.

The following indicators never display normally
when this unit fails.
Within the furnace system,
    the Burner Motor RMP Gauge is 525.
Within the water pump and safety cutoff valve,
    the water pump sound indicator is normal.

The following indicators never display abnormally
when this unit fails.
Within the fire door sight hole,
    the visual combustion check indicator is igniting.

The water pump is a poor suspect since the water pump sound is ok.
You have seen that the combustion is abnormal.
Check the units along the path of the oil and the electrical devices.

Figure 3: Answers to *ConsultRu* by *DIAG-orig*, *DIAG-NLP1* and *DIAG-NLP2*

2.1 *DIAG-NLP1*: Syntactic aggregation

DIAG-NLP1[1] (i) introduces syntactic aggregation (Dalianis, 1996; Huang and Fiedler, 1996; Reape and Mellish, 1998; Shaw, 2002) and what we call *structural* aggregation, namely, grouping parts according to the structure of the system; (ii) generates some referring expressions; (iii) models a few rhetorical relations; and (iv) improves the format of the output.

The middle parts of Figs. 2 and 3 show the revised output produced by *DIAG-NLP1*. E.g., in Fig. 2 the RUs of interest 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 highlights that the *Ignitor Assembly* cannot cause the symptom.

In *DIAG-NLP1*, EXEMPLARS accesses the SNePS Knowledge Representation and Reasoning System (Shapiro, 2000) for static domain information.[2] SNePS makes it easy to recognize structural similarities and use shared structures. Using SNePS, we can examine the dimensional structure of an aggregation and its values to give preference to aggregations with top-level dimensions that have fewer values, to give summary statements when a dimension has many values that are reported on, and to introduce simple text structuring in terms of rhetorical relations, inserting relations like *contrast* and *concession* to highlight distinctions between dimensional values (see Fig. 2, middle).

DIAG-NLP1 uses the GNOME algorithm (Kibble and Power, 2000) to generate referential expressions. Importantly, using SNePS propositions can be treated as discourse entities, added to the discourse model and referred to (see *This is ... caused if ...* in Fig. 2, middle). Information about lexical realization, and choice of referring expression is encoded in the appropriate exemplars.

2.2 *DIAG-NLP2*: functional aggregation

In the interest of rapid prototyping, *DIAG-NLP1* was implemented without the benefit of a corpus study. *DIAG-NLP2* is the empirically grounded version of the feedback generator. We collected 23 tutoring interactions between a student using the DIAG tutor on home heating and two human tutors, for a total of 272 tutor turns, of which 235 in reply to *ConsultRU* and 37 in reply to *ConsultInd* (the type of student query is automatically logged). 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, in tabular form, the information that DIAG would use in generating its advice — the same "fact file" that DIAG gives to *DIAG-NLP1* and *DIAG-NLP2* — and types a response that substitutes for DIAG's. The tutor is presented with this information because we wanted to uncover empirical evidence for aggregation rules in our domain. Although we cannot constrain the tutor to mention only the facts that DIAG would have communicated, we can analyze how the tutor uses the information provided by DIAG.

We developed a coding scheme (Glass et al., 2002) and annotated the data. As the annotation was performed by a single coder, we lack measures of intercoder reliability. Thus, what follows should be taken as observations rather than as rigorous findings – useful observations they clearly are, since

[1] *DIAG-NLP1* actually augments and refines the first feedback generator we created for DIAG, *DIAG-NLP0* (Di Eugenio et al., 2002). *DIAG-NLP0* only covered (i) and (iv).

[2] In DIAG, domain knowledge is hidden and hardly accessible. Thus, in both *DIAG-NLP1* and *DIAG-NLP2* we had to build a small knowledge base that contains domain knowledge.

DIAG-NLP2 is based on these observations and its language fosters the most learning.

Our coding scheme focuses on four areas. Fig. 4 shows examples of some of the tags (the SCM is the System Control Module). Each tag has from one to five additional attributes (not shown) that need to be annotated too.

Domain ontology. We tag objects in the domain with their class *indicator, RU* and their states, denoted by *indication* and *operationality*, respectively.

Tutoring actions. They include (i) *Judgment*. The tutor evaluates what the student did. (ii) *Problem solving*. The tutor suggests the next course of action. (iii) The tutor imparts *Domain Knowledge*.

Aggregation. Objects may be *functional aggregates*, such as *the oil burner*, which is a system component that includes other components; *linguistic aggregates*, which include plurals and conjunctions; or a *summary* over several unspecified indicators or RUs. *Functional/linguistic aggregate* and *summary* tags often co-occur, as shown in Fig. 4.

Relation to DIAG's output. Contrary to all other tags, in this case we annotate the input that DIAG gave the tutor. We tag its portions as *included / excluded / contradicted*, according to how it has been dealt with by the tutor.

Tutors provide explicit problem solving directions in 73% of the replies, and evaluate the student's action in 45% of the replies (clearly, they do both in 28% of the replies, as in Fig. 4). As expected, they are much more concise than DIAG, e.g., they never mention RUs that cannot or are not as likely to cause a certain problem, such as, respectively, the *ignitor assembly* and *the SCM* in Fig. 2.

As regards aggregation, 101 out of 551 RUs, i.e. 18%, are labelled as summary; 38 out of 193 indicators, i.e. 20%, are labelled as summary. These percentages, though seemingly low, represent a considerable amount of aggregation, since in our domain some items have very little in common with others, and hence cannot be aggregated. Further, tutors aggregate parts functionally rather than syntactically. 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*.

Finally, *directness* – an attribute on the *indicator* tag – encodes whether the tutor explicitly talks about the indicator (e.g., *The water temperature gauge reading is low*), or implicitly via the object to which the indicator refers (e.g., *the water is too cold*). 110 out of 193 indicators, i.e. 57%, are marked as *implicit*, 45, i.e. 41%, as *explicit*, and 2% are not marked for directness (the coder was free to leave attributes unmarked). This, and the 137 occurrences of *indication*, prompted us to refer to objects and their states, rather than to indicators (as implemented by Steps 2 in Fig. 5, and 2(b)i, 3(b)i, 3(c)i in Fig. 6, which generate *The combustion is abnormal* and *The water pump sound is OK* in Figs. 2 and 3).

2.3 Feedback Generation in *DIAG-NLP2*

In *DIAG-NLP1* the fact file provided by DIAG is directly processed by EXEMPLARS. In contrast, in *DIAG-NLP2* a planning module manipulates the information before passing it to EXEMPLARS. This module decides which information to include according to the type of query the system is responding to, and produces one or more *Sentence Structure* objects. These are then passed to EXEMPLARS that transforms them into Deep Syntactic Structures. Then, a sentence realizer, RealPro (Lavoie and Rambow, 1997), transforms them into English sentences.

Figs. 5 and 6 show the control flow in *DIAG-NLP2* for feedback generation for *ConsultInd* and *ConsultRU*. Step 3a in Fig. 5 chooses, among all the RUs that DIAG would talk about, only those that would definitely result in the observed symptom. Step 2 in the AGGREGATE procedure in Fig. 5 uses a simple heuristic to decide whether and how to use functional aggregation. For each RU, its possible aggregators and the number n of units it covers are listed in a table (e.g., *electrical devices* covers 4 RUs, *ignitor, photoelectric cell, transformer* and *burner motor*). If a group of REL-RUs contains k units that a certain aggregator *Agg* covers, if $k < \frac{n}{2}$, *Agg* will not be used; if $\frac{n}{2} \leq k < n$, *Agg* preceded by *some of* will be used; if $k = n$, *Agg* will be used.

DIAG-NLP2 does not use SNePS, but a relational database storing relations, such as the ISA hierarchy (e.g., *burner motor* IS-A RU), information about referents of indicators (e.g., *room temperature gauge* REFERS-TO *room*), and correlations between RUs and the indicators they affect.

3 Evaluation

Our empirical evaluation is a three group, between-subject study: one group interacts with *DIAG-orig*,

$[_{judgment}$ $[_{replaceable-unit}$ the ignitor] is a poor suspect] since $[_{indication}$ combustion is working] during startup. The problem is that the SCM is shutting the system off during heating.
$[_{domain-knowledge}$ The SCM reads $[_{summary}$ $[_{linguistic-aggregate}$ input signals from sensors]] and uses the signals to determine how to control the system.]
$[_{problem-solving}$ Check the sensors.]

Figure 4: Examples of a coded tutor reply

```
1. IND ← queried indicator
2. Mention the referent of IND and its state
3. IF IND reads abnormal,
   (a) REL-RUs ← choose relevant RUs
   (b) AGGR-RUs ← AGGREGATE(REL-RUs)
   (c) Suggest to check AGGR-RUs

AGGREGATE(RUs)
1. Partition REL-RUs into subsets by system structure
2. Apply functional aggregation to subsets
```

Figure 5: *DIAG-NLP2*: Feedback generation for *ConsultInd*

```
1. RU ← queried RU
   REL-IND ← indicator associated to RU

2. IF RU warrants suspicion,
   (a) state RU is a suspect
   (b) IF student knows that REL-IND is abnormal
        i. remind him of referent of REL-IND and
           its abnormal state
       ii. suggest to replace RU
   (c) ELSE suggest to check REL-IND

3. ELSE
   (a) state RU is not a suspect
   (b) IF student knows that REL-IND is normal
        i. use referent of REL-IND and its normal state
           to justify judgment
   (c) IF student knows of abnormal indicators OTHER-INDs
        i. remind him of referents of OTHER-INDs
           and their abnormal states
       ii. FOR each OTHER-IND
           A. REL-RUs ← RUs associated with OTHER-IND
           B. AGGR-RUs ← AGGREGATE(REL-RUs)
              ∪ AGGR-RUs
      iii. Suggest to check AGGR-RUs
```

Figure 6: *DIAG-NLP2*: Feedback generation for *ConsultRU*

one with *DIAG-NLP1*, one with *DIAG-NLP2*. The 75 subjects (25 per group) were all science or engineering majors affiliated with our university. Each subject read some short material about home heating, went through one trial problem, then continued through the curriculum on his/her own. The curriculum consisted of three problems of increasing difficulty. As there was no time limit, every student solved every problem. Reading materials and curriculum were identical in the three conditions.

While a subject was interacting with the system, a log was collected 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. We will refer to all the measures that were automatically collected as *performance measures*.

At the end of the experiment, each subject was administered a questionnaire divided into three parts. The first part (the posttest) consists of three questions and tests what the student learned about the domain. The second 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 that the system collects. We expect precision and recall of the replaced RUs to correlate with *transfer*, namely, to predict how well a subject is able to apply what s/he learnt about diagnosing malfunctions to new problems. The third part concerns usability, to be discussed below.

We found that subjects who used *DIAG-NLP2* had significantly higher scores on the posttest, and were significantly more correct (higher precision) in remembering what they did. As regards performance measures, there are no so clear cut results. As regards usability, subjects prefer *DIAG-NLP1*/2 to *DIAG-orig*, however results are mixed as regards which of the two they actually prefer.

In the tables that follow, boldface indicates significant differences, as determined by an analysis of variance performed via ANOVA, followed by post-hoc Tukey tests.

Table 1 reports learning measures, average across the three problems. *DIAG-NLP2* is significantly better as regards PostTest score ($F = 10.359, p = 0.000$), and RU Precision ($F = 4.719, p = 0.012$). Performance on individual questions in the

	DIAG-orig	*DIAG-NLP1*	*DIAG-NLP2*
PostTest	0.72	0.69	**0.90**
RU Precision	0.78	0.70	**0.91**
RU Recall	.53	.47	.40

Table 1: Learning Scores

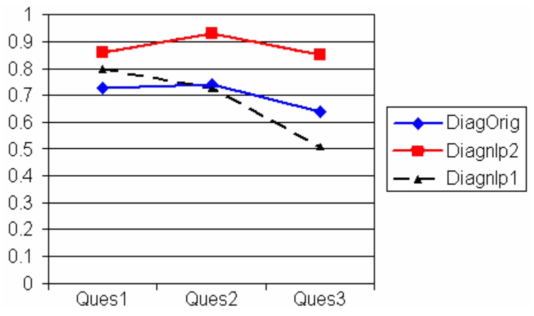

Figure 7: Scores on PostTest questions

PostTest[3] is illustrated in Fig. 7. Scores in *DIAG-NLP2* are always higher, significantly so on questions 2 and 3 ($F = 8.481, p = 0.000$, and $F = 7.909, p = 0.001$), and marginally so on question 1 ($F = 2.774, p = 0.069$).[4]

	D-Orig	*D-NLP1*	*D-NLP2*
Total Time	30'17"	28'34"	34'53"
RU Replacements	8.88	11.12	11.36
ConsultInd	22.16	**6.92**	28.16
Avg. Reading Time	8"	14"	**2"**
ConsultRU	63.52	45.68	52.12
Avg. Reading Time	5"	4"	5"

Table 2: Performance Measures

Table 2 reports performance measures, cumulative across the three problems, other than average reading times. Subjects don't differ significantly in the time they spend solving the problems, or in the number of *RU replacements* they perform. DIAG's assumption (known to the subjects) is that there is only one broken RU per problem, but the simulation allows subjects to replace as many as they want without any penalty before they come to the correct solution. The trend on *RU replacements* is opposite what we would have hoped for: when repairing a real system, replacing parts that are working should clearly be kept to a minimum, and subjects replace fewer parts in *DIAG-orig*.

The next four entries in Table 2 report the number of queries that subjects ask, and the average time it takes subjects to read the feedback. The subjects ask significantly fewer *ConsultInd* in *DIAG-NLP1* ($F = 8.905, p = 0.000$), and take significantly less time reading *ConsultInd* feedback in *DIAG-NLP2* ($F = 15.266, p = 0.000$). The latter result is not surprising, since the feedback in *DIAG-NLP2* is much shorter than in *DIAG-orig* and *DIAG-NLP1*. Neither the reason not the significance of subjects asking many fewer *ConsultInd* of *DIAG-NLP1* are apparent to us – it happens for *ConsultRU* as well, to a lesser, not significant degree.

We also collected usability measures. Although these are not usually reported in ITS evaluations, in a real setting students should be more willing to sit down with a system that they perceive as more friendly and usable. Subjects rate the system along four dimensions on a five point scale: clarity, usefulness, repetitiveness, and whether it ever misled them (the scale is appropriately arranged: the highest clarity but the lowest repetitiveness receive 5 points). There are no significant differences on individual dimensions. Cumulatively, *DIAG-NLP2* (at 15.08) slightly outperforms the other two (*DIAG-orig* at 14.68 and *DIAG-NLP1* at 14.32), however, the difference is not significant (highest possible rating is 20 points).

	prefer	neutral	disprefer
DIAG-NLP1 to *DIAG-orig*	28	5	17
DIAG-NLP2 to *DIAG-orig*	34	1	15
DIAG-NLP2 to *DIAG-NLP1*	24	1	25

Table 3: User preferences among the three systems

	prefer	neutral	disprefer
Consult Ind.	8	1	16
Consult RU	16	0	9

Table 4: *DIAG-NLP2* versus *DIAG-NLP1*

	natural	concise	clear	contentful
DIAG-NLP1	4	8	10	23
DIAG-NLP2	16	8	11	12

Table 5: Reasons for system preference

Finally,[5] on paper, subjects compare two pairs of versions of feedback: in each pair, the first feedback

[3]The three questions are: 1. Describe the main subsystems of the furnace. 2. What is the purpose of (a) the oil pump (b) the system control module? 3. Assume the photoelectric cell is covered with enough soot that it could not detect combustion. What impact would this have on the system?

[4]The PostTest was scored by one of the authors, following written guidelines.

[5]Subjects can also add free-form comments. Only few did

is generated by the system they just worked with, the second is generated by one of the other two systems. Subjects say which version they prefer, and why (they can judge the system along one or more of four dimensions: natural, concise, clear, contentful). The first two lines in Table 3 show that subjects prefer the NLP systems to *DIAG-orig* (marginally significant, $\chi^2 = 9.49, p < 0.1$). *DIAG-NLP1* and *DIAG-NLP2* receive the same number of preferences; however, a more detailed analysis (Table 4) shows that subjects prefer *DIAG-NLP1* for feedback to *ConsultInd*, but *DIAG-NLP2* for feedback to *ConsultRu* (marginally significant, $\chi^2 = 5.6, p < 0.1$). Finally, subjects find *DIAG-NLP2* more natural, but *DIAG-NLP1* more contentful (Table 5, $\chi^2 = 10.66, p < 0.025$).

4 Discussion and conclusions

Our work touches on three issues: aggregation, evaluation of NLG systems, and the role of NL interfaces for ITSs.

In much work on aggregation (Huang and Fiedler, 1996; Horacek, 2002), aggregation rules and heuristics are shown to be plausible, but are not based on any hard evidence. Even where corpus work is used (Dalianis, 1996; Harvey and Carberry, 1998; Shaw, 2002), the results are not completely convincing because we do not know for certain the content to be communicated from which these texts supposedly have been aggregated. Therefore, positing empirically based rules is guesswork at best. Our data collection attempts at providing a more solid empirical base for aggregation rules; we found that tutors exclude significant amounts of factual information, and use high degrees of aggregation based on functionality. As a consequence, while part of our rules implement standard types of aggregation, such as conjunction via shared participants, we also introduced functional aggregation (see *conceptual* aggregation (Reape and Mellish, 1998)).

As regards evaluation, NLG 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, i.e., measuring how well the users of the system perform on the task at hand (Young, 1999; Carenini and Moore, 2000; Reiter et al., 2003). The latter kind of studies generally contrast different interventions, i.e. a baseline that does not use NLG and one or more variations obtained by parameterizing the NLG system. However, the evaluation does not focus on a specific component of the NLG process, as we did here for aggregation.

Regarding the role of NL interfaces for ITSs, only very recently have the first few results become available, to show that first of all, students do learn when interacting in NL with an ITS (Litman et al., 2004; Graesser et al., 2005). However, there are very few studies like ours, that evaluate specific features of the NL interaction, e.g. see (Litman et al., 2004). In our case, we did find that different features of the NL feedback impact learning. Although we contend that this effect is due to functional aggregation, the feedback in *DIAG-NLP2* changed along other dimensions, mainly using referents of indicators instead of indicators, and being more strongly directive in suggesting what to do next. Of course, we cannot argue that our best NL generator is equivalent to a human tutor – e.g., dividing the number of *ConsultRU* and *ConsultInd* reported in Sec. 2.2 by the number of dialogues shows that students ask about 10 *ConsultRus* and 1.5 *ConsultInd* per dialogue when interacting with a human, many fewer than those they pose to the ITSs (cf. Table 2) (regrettably we did not administer a PostTest to students in the human data collection). We further discuss the implications of our results for NL interfaces for ITSs in a companion paper (Di Eugenio et al., 2005).

The DIAG project has come to a close. We are satisfied that we demonstrated that even not overly sophisticated NL feedback can make a difference; however, the fact that *DIAG-NLP2* has the best language and engenders the most learning prompts us to explore more complex language interactions. We are pursuing new exciting directions in a new domain, that of basic data structures and algorithms. We are investigating what distinguishes expert from novice tutors, and we will implement our findings in an ITS that tutors in this domain.

Acknowledgments. This work is supported by the Office of Naval Research (awards N00014-99-1-0930 and N00014-00-1-0640), and in part by the National Science Foundation (award IIS 0133123). We are grateful to CoGenTex Inc. for making EXEMPLARS and RealPro available to us.

so, and the distribution of topics and of evaluations is too broad to be telling.

References

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

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. Trolio. 2002. The DIAG experiments: Natural Language Generation for Intelligent Tutoring Systems. In *INLG02, The Third International Natural Language Generation Conference*, pages 120–127.

Barbara Di Eugenio, Davide Fossati, Dan Yu, Susan Haller, and Michael Glass. 2005. Natural language generation for intelligent tutoring systems: a case study. In *AIED 2005, the 12th International Conference on Artificial Intelligence in Education*.

M. W. Evens, J. Spitkovsky, P. Boyle, J. A. Michael, and A. A. Rovick. 1993. Synthesizing tutorial dialogues. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, pages 137–140.

Michael Glass, Heena Raval, Barbara Di Eugenio, and Maarika Traat. 2002. The DIAG-NLP dialogues: coding manual. Technical Report UIC-CS 02-03, University of Illinois - Chicago.

A.C. Graesser, N. Person, Z. Lu, M.G. Jeon, and B. McDaniel. 2005. Learning while holding a conversation with a computer. In L. PytlikZillig, M. Bodvarsson, and R. Brunin, editors, *Technology-based education: Bringing researchers and practitioners together*. Information Age Publishing.

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.

Helmut Horacek. 2002. Aggregation with strong regularities and alternatives. In *International Conference on Natural Language Generation*.

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.

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

Benoît Lavoie and Owen Rambow. 1997. A fast and portable realizer for text generation systems. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*.

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

D. J. Litman, C. P. Rosé, K. Forbes-Riley, K. VanLehn, D. Bhembe, and S. Silliman. 2004. Spoken versus typed human and computer dialogue tutoring. In *Proceedings of the Seventh International Conference on Intelligent Tutoring Systems*, Maceio, Brazil.

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

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

Ehud Reiter, Roma Robertson, and Liesl Osman. 2003. Lessons from a failure: Generating tailored smoking cessation letters. *Artificial Intelligence*, 144:41–58.

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

James Shaw. 2002. A corpus-based analysis for the ordering of clause aggregation operators. In *COLING02, Proceedings of the 19th International Conference on Computational Linguistics*.

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

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.

R. Michael Young. 1999. Using Grice's maxim of quantity to select the content of plan descriptions. *Artificial Intelligence*, 115:215–256.