Development and Evaluation of NL interfaces in a Small Shop[*]

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

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

Michael Glass[†]
Math & CS
Valparaiso University
Valparaiso, IN, 46383, USA
Michael.Glass@valpo.edu

Abstract

The standard development of a dialogue system today involves the following steps: corpus collection and analysis, system development guided by corpus analysis, and finally, rigorous evaluation. Often, evaluation may involve more than one version of the system, for example when it is desirable to show the effect of system parameters that differ from one version to another. In this paper, we discuss the difficulties that small research groups face in pursuing the development of dialogue systems. The primary difficulties are the lack of adequate resources and the excessive amount of time it takes to see the systems through to a meaningful evaluation. As a case in point, we discuss our development and evaluation of a natural language generation component to improve the feedback provided by an interactive tutoring system. Our goal has been to use relatively inexpensive text structuring techniques to make aggregate content more fluent and comprehensible.

Introduction

Early on, developers of dialogue interfaces used to work mainly on the basis of their own intuitions. After completing a system, it was evaluated rather informally. This could be just checking whether or not the system corresponded to specifications, or it could take the form of informal user studies. The current development standard for a dialogue system has the following steps: 1) a corpus is collected and analyzed, 2) a system is developed according to the corpus findings, and 3) a rigorous evaluation is conducted. Often the evaluation may involve more than one version of the system, if it is desirable to show the effect of system parameters that differ from one version to another (Young 1997; Carenini & Moore 2000; Di Eugenio, Glass, & Trolio 2002).

The current standard definitely transforms system development into a rigorous scientific enterprise. The strengths and weaknesses of the evaluated system are clearly assessed; the results are replicable, and they guide further system development. However, from the point of view of the individual researcher, the current standard has transformed an already time consuming endeavor (building systems) into an endeavor that can take three times as long. Building a system can take one or two person-years. Collecting and analyzing a corpus may take as long, and evaluation may take at least half as long. This is particularly problematic for small natural language research groups - groups with one lead researcher, one or two graduate students or a postdoctoral fellow, and possibly one or two external collaborators.

We would like to see this issue addressed at the symposium with the hope that good suggestions may arise on how to shorten the development cycle without compromising its rigor. We contribute some of our own suggestions for doing so in this paper. But first, as a case study, we describe our development and evaluation of a natural language generation component to improve the feedback provided by an interactive tutoring system.

From the onset of this project, our goal has been to use relatively inexpensive domain-independent text structuring techniques to make aggregate content more fluent and comprehensible. We were interested in understanding which features of language feedback affect student learning. Towards this end, we developed three different prototypes. The first two prototypes differ in the presentation of the language feedback they provide. Prototype 1 (DIAG-NLP1) has been evaluated and reported on (Di Eugenio, Glass, & Trolio 2002); prototype 2 (DIAG-NLP2) is fully implemented (Haller & Di Eugenio 2002) and recently underwent evaluation. In contrast to the first two prototypes, prototype 3 (DIAG-NLP3) presents the same content but at a higher level of abstraction. It is almost fully implemented and will undergo evaluation as soon as it is ready. Moreover, a full data collection and analysis has been conducted; the findings from the data were used to develop prototype DIAG-NLP3, but not the other prototypes. The project started in Fall 2000 and to be completed, it will last through Summer 2003.

In the following sections, we describe the project in greater detail: the three protot ypes, the data collection, and the results of evaluating DIAG-NLP1. Then we return to our discussion of the development cycle, and based on our experiences with this project, we offer some suggestions on how

[*]This work is supported by grants N00014-99-1-0930 and N00014-00-1-0640 from the Office of Naval Research. We are grateful to CoGenTex Inc., in particular to Mike White and Benoît Lavoie, for making EXEMPLARS and RealPro available to us.

[†]This work was performed while Michael Glass was a postdoctoral fellow at the University of Illinois - Chicago.

it might be made more efficient.

The DIAG-NLP project

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

The first prototype: DIAG-NLP1

We built DIAG-NLP1 with the aim of rapidly improving DIAG’s feedback mechanism. Our two main goals were 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 & Fiedler 1996; Shaw 1998; Reape & 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. We call these groupings the *system* and *certainty dimensions* of the aggregation, and the actual values that we aggregate units are the *system* and *certainty dimension values*. 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 & Caldwell 1998), an object-oriented, rule based generator. It mixes template-style text planning with a more sophisticated type of text planning based on dynamic dispatch. 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.

After a student query, DIAG collects the content that it needs to communicate to the student, and writes it to a text file that is passed to EXEMPLARS. EXEMPLARS performs 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 a text window.

In DIAG-NLP1, morphology, lexical realization and referring expression generation were all directly encoded in the appropriate exemplars.

Evaluation of DIAG-NLP1. Our empirical evaluation of DIAG-NLP1 is a between-subject study: one group interacts with DIAG-orig and the other with DIAG-NLP1. The 34 subjects (17 per group) were all science or engineering majors affiliated with the University of Illinois at Chicago. Each subject read some short material about home heating, went through the first problem as a trial run, and 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 given a questionnaire.

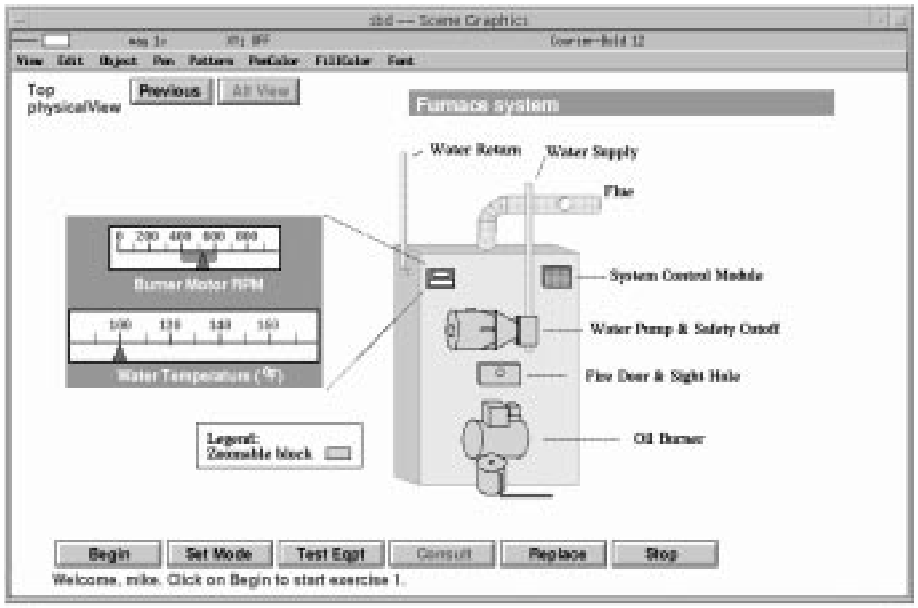
We collected a log for each subject that included for each problem: whether the problem was solved; total time, and time spent reading feedback; how many and which indicators and RUs the subject consults DIAG about; and how many and which RUs the subject replaces.

The questionnaire is divided into three parts. The first part tests the subject’s understanding of the domain. Because these questions are open ended, it was scored like 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 that the system collects. In Table 2, we report the F-measure ($\frac{(\beta^2+1)PR}{\beta^2 P+R}$, with $\beta = 1$) that smooths precision and recall.

Results

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

Although individually all but one or two measures favor DIAG-NLP1, differences are not statistically significant. *Indicator consultations* comes closest to significance with a non-significant trend in favor of DIAG-NLP1 (Mann-Whitney test, *U=98, p=0.11*).



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

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

	DIAG-orig	DIAG-NLP1
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-NLP1
Essay score	81/100	83/100
RU recollection	.72	.63

Table 2: Learning and recollection measures

	DIAG-orig	DIAG-NLP1
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

	DIAG-orig	DIAG-NLP1
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

We therefore apply the binomial cumulative distribution function (BCDF), or one-tailed Sign Test, to assess whether DIAG-NLP1 is better than DIAG-orig. This test measures the likelihood that DIAG-NLP1 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, Glass, & Scott 2002) discusses the BCDF further).

Table 4 combines the independent measures from Tables 1, 2, and 3, showing which condition was more successful. The result shows 9/10 (or 8/10) wins for DIAG-NLP1. Since one measure was tied, we report two sets of probabilities assuming that the tied measure favored DIAG-orig or DIAG-NLP1 respectively.

The probability of 9/10 (or 8/10) successes for DIAG-NLP1 under the null hypothesis is $p = 0.01$ (or 0.054), showing a significant (or marginally significant) win for DIAG-NLP1. 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-NLP1.

Had we followed the customary practice of discarding the tied measure (Siegel & Castellan 1988),[1] DIAG-NLP1 would win 8/9, $p = 0.02$, or 7/8, $p = 0.035$ (depending on the inclusion of *Total Time*), which are both significant.

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

```
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: DIAG-orig (top) versus DIAG-NLP1 (bottom) replies to the same *Consult Indicator* query

We can then conclude that the better measures for DIAG-NLP1, albeit individually not statistically significant, cumulatively show that DIAG-NLP1 outperforms DIAG-orig.

The second prototype: DIAG-NLP2

DIAG-NLP2 arose from the following consideration: although DIAG-NLP1 aggregates content by system and certainty, it still produces repetitive feedback. At the same time, DIAG-NLP1 was faster to build than a full fledged NL generator. We were interested in whether we could maintain the basic architecture of DIAG-NLP1 and at the same time make the aggregate content significantly more fluent and comprehensible. In DIAG-NLP2, specific rhetorical relations such as *contrast* and *concession* were introduced based on the data itself in a bottom-up fashion rather than being planned top-down by the discourse planner. Our aim was to 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 techniques that work locally. DIAG-NLP2 generates text by coupling EXEMPLARS 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.

Figure 3 gives another response generated by DIAG-NLP1 (top). The response aggregates information about RUs first by 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 observed symptom ("always", "often", "sometimes", "never"). Although the aggregation imposes an organization on the information, it still fails to make that organization quickly understandable. For example, it is easy to overlook the transition between certainty values "never" and "always" in going from 4-9 to 10.

Figure 4 gives the same response as generated by DIAG-NLP2. Note that the aggregate structure is still the same. However, the *contrast* rhetorical relation (Mann & Thompson 1988) is used between units that "never" (lines 7-8) and units that "always" (lines 9-10) cause the indication begin discussed.

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 structures of the aggregation and their values using inexpensive techniques for text structuring and for referential expression generation, a more robust

```
 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 indication
         when it fails.
```

Figure 3: Another response by DIAG-NLP1

```
 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,
 5       this is sometimes caused if
 6       the system control module has failed.

 7   Within the oil burner,
 8       this is never caused if the ignitor assembly has failed.
 9       In contrast, this is always caused if
10       the burner motor, oil filter, oil pump, oil supply valve, or oil nozzle has failed.
```

Figure 4: The same response as generated by DIAG-NLP2

knowledge representation of the domain, and a small amount of lexical information.

Unlike DIAG-NLP1, in DIAG-NLP2, we prefer aggregations that have fewer dimensional values as the first dimension to present, and we generate referential expressions, including references to whole propositions.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.

Whereas DIAG-NLP1 generated referential expressions ad hoc, in DIAG-NLP2 we implemented the GNOME algorithm to generate referential expressions (Kibble & Power 2000). It uses insights from centering (Grosz, Joshi, & Weinstein 1995) and from theories of salience. Importantly, SNePS 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 line 2, Figure 4. *This* refers to the entire clause in line 1.

We have just run a user study to evaluate DIAG-NLP2, but we have not concluded the data analysis yet.

The third prototype: DIAG-NLP3

DIAG-NLP3 is being developed on the basis of a corpus study. We conducted a constrained data collection to uncover empirical evidence for the EXEMPLARS rules we implemented in DIAG-NLP1 and DIAG-NLP2. 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 & Fiedler 1996; Shaw 1998), in which aggregation rules and heuristics are plausible, but are not based on any hard evidence.

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. Communication is typed but does not have "Wizard of Oz" secrecy. 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-NLP1 and DIAG-NLP2. 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 developed a coding scheme (Glass *et al.* 2002) and annotated the data (all of the data has been annotated by one coder and at least half of it by a second coder). The difference between DIAG response style and human tutors is striking. Specifically:

- in 71% of the cases, tutors teach: they don't just paraphrase the knowledge that DIAG presents to them, but
  - in 21% of the cases, they evaluate what the student did or asked about *The infrared sensor is a good thing to think about*;
  - in 50% of the cases, they suggest the next course of action: *Check the sensors*.
- Tutors do not usually provide lists of replaceable units, abstracting away from individual parts as possible (see below for examples)
- Tutors frequently omit mention of parts that cannot be causing the problem.

Human tutors eschew syntactic aggregation of part lists and instead describe functional aggregations of parts. This is consistent with Paris' work on the TAILOR system for describing patented systems. For a reader with little or no knowledge of the domain, TAILOR was designed to plan a description of an object in terms of its function and the function of its parts (Paris 1988). On the one hand, these results lend support to the rule that groups parts according to the system hierarchical structure that we implemented in DIAG-NLP1 and DIAG-NLP2. However, the aggregations favored by human tutors go one step further: they are at a functional level. 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* , as *the path of the oil flow*, or as *the units that the oil would normally flow through*.

On the basis of this, we are developing a third prototype that mirrors the findings from the corpus. The NLG architecture has been completely redesigned, so as to make it more flexible. Moreover, the NLG system is now coupled to RealPro (Lavoie & Rambow 1997) that performs syntactic and lexical realization. RealPro is a text generation ”engine” that performs syntactic realization. RealPro provides a grammar rule engine that can generate text from sophisticated, multi-level linguistic representations. The abstraction it provides makes it easy to generate many syntactic variants of the same semantic content on demand.

DIAG-NLP3 has been fully implemented with respect to feedback on indicators, but still needs development on feedback for replaceable units. Figure 5 shows the response generated by DIAG-NLP3 in response to the same query to which DIAG-ORIG and DIAG-NLP1 respond in Figure 2. Note how the long list of parts has been replaced by an abstract description; moreover, there is no mention of parts who are unlikely to cause the problem (i.e., the system control module and the ignitor assembly in Figure 2). We believe that DIAG-NLP3 generates the clearest, most natural sounding and least repetitive responses of all prototypes, even if we have weak anecdotal evidence to the contrary.[2] While evaluating DIAG-NLP2, we gave subjects one pair of responses, the one by DIAG-NLP3 shown in Figure 5, and the corresponding one generated by DIAG-NLP2. We then asked them to express their preference between the two. Thirteeen subjects preferred DIAG-NLP2, nine DIAG-NLP2, and one expressed no preference. Of course, only a formal evaluation can really tell us whether DIAG-NLP3's responses are better than the other prototypes'. We will conduct it as soon as DIAG-NLP3 is completed.

Towards more efficient development and evaluation in the small shop

Now that we have provided a sense for the project in general, we will detail the associated timelines. DIAG-NLP1 took six months to develop plus three months to evaluate (this comprises both running subjects and analyzing the data). DIAG-NLP2 exploited tools developed for DIAG-NLP1, therefore it only took two full-time person months to develop, plus another two to evaluate. The data collection and analysis that we described took at least one year. DIAG-NLP3 has taken at least 4 months so far, it will probably take at least another two to complete, and another two or three to evaluate. Each of the systems / evaluations / data collection was done by one person (graduate student or consultant or postdoc), under the supervision of the first author, and with occasional help from one other graduate student.

To put this timeline in perspective, note that in all these cases, system development was simplified by the constrained task that the system has to perform, and by using software components such as EXEMPLARS, RealPro and SNePS. Using existing software is a tremendous help, although of course at a cost, given a substantial learning curve is incurred. Clearly, developing a full fledged dialogue system would require far more extensive resources: it is not an enterprise in which a single researcher with one or two graduate students can really embark. What are the possible approaches the natural language software development and testing in this context? We put forth a few ideas with the hope more will come forth at the symposium. Although much of what we discuss is “wishful thinking”, our outlook on these issues has become more optimistic since submitting the original paper just a few months back. This is because we found evidence that developing standards for annotation may not be utopic after all, and that training across corpora may be highly beneficial for system development (Chen *et al.* 2002). More details in what follows.

Data collection / analysis. The only answer seems to be to develop repositories of tagged corpora. Special interest groups such as ACL-SigDIAL (http://www.sigdial.org)

[2]Supporting one anonymous reviewer's observation that s/he would prefer DIAG-NLP1's version to DIAG-NLP3's.

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

Figure 5: Response generated by DIAG-NLP3 (contrast with the response by DIAG-NLP1 in Figure 2)

do collect annotated corpora. But even when such corpora are available, the annotation always suffers from being special purpose, since everybody including ourselves is collecting their data for some very specific purpose. To us, the answer is that standards for annotation should be developed so that corpora can be effectively shared. The common objection to such an approach, voiced for example by one of the reviewers of this paper, is that a standard is impossible because annotation depends on *basic beliefs about the definition and importance of various discourse phenomena. Discourse has not yet become standardized the way syntax has.* This is certainly true, and it may be a reason why the Discourse Resource Initiative (http://www.georgetown.edu/luperfoy/Discourse-Treebank/dri-home.html) is stalled and it is not clear whether it will ever resume. The DRI tried to develop at least a skeleton of a standard coding scheme for dialogue and discourse (Allen & Core 1997). A related effort was MATE (the Multilevel Annotation, Tools Engineering european project, http://mate.nis.sdu.dk/), which also devoted considerable effort to both a standard coding scheme and tools to support annotations. As far as we know, the tool (the MATE workbench (Dybkjær *et al.* 2000)) has generated quite some interest, but not the annotation framework per se.

On the bright side though, it is of note that the DRI draft annotation scheme (Allen & Core 1997) has been used by a variety of projects as the foundation of their annotation schemes. Among them: SWBD-DAMSL (Jurafsky, Shriberg, & Biasca 1997) developed for a corpus of generic phone conversations; COCONUT (Di Eugenio, Jordan, & Pylkkänen 1998), used to code collaborative computer-mediated dialogues; the schemes used on the Italian spoken corpora ADAM and AVIP (Soria 2002); the scheme developed for the French corpus AMITIÉS (Hardy *et al.* 2002) which consists of customer calls to financial call centers.

Moreover, a more fruitful approach to developing annotation standards may be focusing on *reference tasks*, as recently explored at the ISLE workshop on Dialogue Tagging for Multi-modal Human Computer Interaction. The idea is to identify specific reference tasks to which a tag set is relevant, rather than developing a tag set that is supposed to apply across any discourse / dialogue genres (http://www.research.att.com/ walker/isle-dtag-wrk/).

Finally, there are efforts to automatically or at least semi-automatically label corpora for dialogue acts, given a small manually annotated portion of the data, see e.g. (Walker, Passonneau, & Boland 2001). It remains to be seen how successful and portable across domains such taggers will be.

System development. Again, one answer is to develop repositories of reusable software components. For example, we used NL generation components provided by CoGenTex Inc. In fact, it appears to us that at the moment it is easier to share software than corpora. The most ambitious embodiment of this approach is the DARPA communicator, which supports a *plug and play* architecture via the Galaxy Communicator. This is a *a distributed, message-based, hub-and-spoke infrastructure optimized for constructing spoken dialogue systems* (http://fofoca.mitre.org/). The idea is that one could plug in one's own components into the underlying software architecture. Several sites are adopting this approach in developing their dialogue systems. A plug-and-play architecture if indeed usable would definitely help, although it would not solve the basic problem: how to develop the individual components to plug in.

Reusable *trainable* software components seem to be the answer. However, we go back to the original problem: how to build the annotated corpora necessary to train them on. Recent work opens some exciting possibility in this regard. (Chen *et al.* 2002) provides some evidence that a system can be successfully trained on a corpus which was developed and annotated for a *different* application in a *different* domain. (Chen *et al.* 2002) shows that a surface realizer trained on a small partially parsed in-domain corpus (the domain is travel planning) performs better than if trained on a larger corpus such as the Penn TreeBank (Marcus, Santorini, & Marcinkiewicz 1993). However, they also show that if the realizer is trained using automatically extracted grammars, a larger, out-of-domain corpus such as the Penn TreeBank is more beneficial than the small in-domain corpus. Whether these results can be extended to other domains and other components of a dialogue system remains to be seen. However, it is definitely an exciting avenue to explore.

Evaluation. How can we save time and avoid reduplication of effort? It should become common practice to contract out evaluation to external consultants. This, of course, requires funds to pay them, so it would not help small research groups. One solution may be to evaluate software against additional corpora instead of actual users. Evaluation on a corpus cannot answer questions pertaining to cognitive effects, but can at least show whether the system performs up to specifications. Of course, such an evaluation requires a corpus annotated in an appropriate way, so we go back to the issues raised earlier.

Collaboration. A general approach is to develop collaborations among small (and large!) research groups and assign different groups responsibility for corpora collection and analysis, system development, and evaluation. The advantage is that some subtasks could be pipelined (for example, corpora analysis could be ongoing with early system development). The disadvantage is that is would require close coordination among these groups which might be overly optimistic.

References

Allen, J., and Core, M. 1997. Draft of DAMSL: Dialog act markup in several layers. Coding scheme developed by the participants at two Discourse Tagging Workshops, University of Pennsylvania March 1996, and Schloß Dagstuhl, February 1997.

Carenini, G., and Moore, J. D. 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*.

Chen, J.; Bangalore, S.; Rambow, O.; and Walker, M. A. 2002. Towards automatic generation of natural language generation systems. In *COLING02, Proceedings of the Ninteenth International Conference on Computational Linguistics*.

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 Scott, M. J. 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*.

Di Eugenio, B.; Glass, M.; and Trolio, M. J. 2002. The DIAG experiments: Natural Language Generation for Intelligent Tutoring Systems. In *INLG02, The Third International Natural Language Generation Conference*, 120–127.

Di Eugenio, B.; Jordan, P. W.; and Pylkkänen, L. 1998. The COCONUT project: Dialogue annotation manual. Technical Report ISP 98-1, University of Pittsburgh. Available at http://www.isp.pitt.edu/~intgen/research-papers.

Dybkjær, L.; Møller, M. B.; Bernsen, N. O.; Olsen, M.; and Schiffrin, A. 2000. Annotating communication problems using the MATE workbench. In *LREC2000, Proceedings of the Second International Conference on Language Resources and Evaluation*, 1557–1564.

Glass, M.; Raval, H.; Di Eugenio, B.; and Traat, M. 2002. The DIAG-NLP dialogues: coding manual. Technical Report UIC-CS 02-03, University of Illinois - Chicago.

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

Haller, S., and Di Eugenio, B. 2002. Text structuring to improve the presentation of aggregated content. Technical Report UIC-CS-02-02, University of Illinois - Chicago.

Hardy, H.; Baker, K.; Devillers, L.; Lamel, L.; Rosset, S.; Strzalkowski, T.; Ursu, C.; and Webb, N. 2002. Multi-layer dialogue annotation for automated multilingual customer service. In *ISLE Workshop: Dialogue Tagging for Multi-Modal Human Computer Interaction*.

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

Jurafsky, D.; Shriberg, E.; and Biasca, D. 1997. Switchboard SWBD-DAMSL Shallow-Discourse-Function Annotation Coders Manual, Draft 13. Technical Report 97-02, University of Colorado, Boulder. Institute of Cognitive Science.

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

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

Mann, W. C., and Thompson, S. 1988. Rhetorical Structure Theory: toward a Functional Theory of Text Organization. *Text* 8(3):243–281.

Marcus, M.; Santorini, B.; and Marcinkiewicz, M. A. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics* 19(2):313–330.

Paris, C. L. 1988. Tailoring object descriptions to the user's level of expertise. *Computational Linguistics* 14(3):64–78.

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

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.

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

Soria, C. 2002. Dialogue tagging for general purposes: the *ADAM* and *AVIP* projects. In *ISLE Workshop: Dialogue Tagging for Multi-Modal Human Computer Interaction*.

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

Walker, M. A.; Passonneau, R.; and Boland, J. E. 2001. Qualitative and quantitative evaluation of DARPA communicator dialogue systems. In *ACL01, Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*.

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*, 266–275.

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