Evaluating a Trainable Sentence Planner for a Spoken Dialogue Travel System

Owen C. Rambow and Monica Rogati and Marilyn A. Walker
AT&T Shannon Labs
Carnegie Mellon University
AT&T Shannon Labs

Paper ID: ACL-2001-0315

Keywords: evaluation, natural language generation, sentence planning, machine learning, spoken dialogue systems

Contact Author: Marilyn A. Walker (walker@research.att.com)

Under consideration for other conferences (specify)? none

Abstract

This paper reports on the evaluation of a trainable sentence planner. We compare a trainable sentence planner with a currently operational hand-crafted template-based generator, two rule-based generators and two baseline generators. We show that the trainable sentence planner performs as well as the hand-crafted generator and better than the rule-based and baseline generators.

Evaluating a Trainable Sentence Planner for a Spoken Dialogue Travel System

Paper-ID: ACL-2001-0315

Abstract

Recently, several techniques for *automatically training* modules of a natural language generator have been proposed. While the engineering benefits of trainable approaches appear obvious, a fundamental concern is whether the quality of system utterances produced with trainable components can compete with hand-crafted template-based systems or with rule-based approaches. In this paper we evaluate the quality of the output of a trainable sentence planner for a spoken dialogue system in the travel planning domain. We directly compare our trainable sentence planner to the hand-crafted template-based generation component of the existing system. In order to perform an exhaustive comparison, we also evaluate two rule-based sentence-planners and two baseline sentence-planners. We show that the trainable sentence planner performs better than both rule-based systems and as well as the hand-crafted template-based system, and that these four systems outperform the baseline sentence planners.

1 Introduction

The past several years have seen a large increase in commercial spoken dialog systems. These systems typically utilize system-initiative dialog strategies, with system utterances highly scripted for style and register and recorded by voice talent. However several factors argue against the continued use of these simple techniques for producing the system side of the conversation. First, the quality of text-to-speech systems has improved to the point of being a viable alternative to pre-recorded prompts. Second, there is a perceived need for spoken dialog systems to be more flexible and support user initiative, but this also requires greater flexibility for system utterance generation. Finally, dialog systems that support complex planning are being developed, and these are likely to require more sophisticated system output.

As we move away from systems with pre-recorded prompts, there are two possible approaches to producing system utterances. The first is TEMPLATE-BASED generation, in which system utterances are produced from hand-crafted string templates with variables that are instantiated by the dialog manager. Most current research systems use template-based generation because it is conceptually fairly easy to produce high quality output that is specific to each dialog situation. However, while little or no linguistic training is needed to write templates, it is a tedious and time-consuming task: one or more templates must be written for each combination of goals and discourse contexts, and linguistic issues such as subject-verb agreement and determiner-noun agreement must be encoded in an ad-hoc fashion each time the situation arises. Furthermore, maintenance of the collection of templates becomes a software engineering problem as the complexity of the dialog system increases.[1]

The second approach is NATURAL LANGUAGE GENERATION (NLG), which customarily divides the generation process into three modules (Rambow and Korelsky, 1992): (1) Text Planning, (2) Sentence Planning, and (3) Surface Realization. NLG promises portability across application domains and dialog situations by focusing on the development of rules for each generation module that are general and domain-independent. However, the quality of the output for a particular domain, or a particular situation in a dialog, may be inferior to that of a template-based system without considerable investment in domain-specific rules or domain-tuning of general rules. Furthermore, since rule-based systems use sophisticated linguistic representations, this handcrafting requires linguistic knowledge.

Recently, several different techniques for *automatically training* different modules of an NLG system have been proposed. These hold the promise that the complex step of customizing NLG systems by hand can be automated, while avoiding the need for tedious hand-crafting of templates. While the engineering benefits of trainable approaches appear obvious, a fundamental concern is whether the quality of utterances produced with trainable components can compete with either hand-crafted template-based systems or with rule-based approaches.

In this paper we address this question by evaluat-

[1] Although we are not aware of any software engineering studies of template development and maintenance, this claim is supported by abundant anecdotal evidence.

System1:	Welcome.... What airport would you like to fly out of?
User2:	I need to go to Dallas.
System3:	Flying to Dallas. What departure airport was that?
User4:	from Newark on September the 1st.
System5:	Flying from Newark to Dallas, Leaving on the 1st of September, And what time did you want to leave?

Figure 1: Dialog D1 with our COMMUNICATOR travel planning system

ing the quality of the output of a trainable sentence planner for a spoken dialog system in the travel planning domain. Because the spoken dialog system is a working DARPA COMMUNICATOR system that participated in the June 2000 DARPA evaluation, we can directly compare the TRAINABLE system to the hand-crafted TEMPLATE-BASED generation component of the current system. In order to perform an exhaustive comparison, we also implemented several RULE-BASED SENTENCE-PLANNERS and several BASELINE SENTENCE-PLANNERS. One baseline, which we call NO AGGREGRATION, simply produces a single sentence for each communicative goal. Another baseline, which we call RANDOM, randomly makes decisions about how to combine communicative goals into sentences. We directly compare the output of these different approaches in an evaluation experiment in which we ask 60 human subjects to compare and judge the quality of the output by rating each system's output on a scale of 1 to 5.

The experimental design is described in section 2. The sentence planners used in the evaluation are described in section 3. In section 4 we present our results. We show that the trainable sentence planner performs better than both rule-based systems and as well as the hand-crafted template-based system. These four systems outperform the baseline sentence planners. Section 5 summarizes our results and discusses related and future work.

2 Experimental Context and Design

Our research concerns developing and evaluating a portable generation component for our mixed-initiative DARPA COMMUNICATOR system. Consider the required generation capabilities such a system, as illustrated in Figure 1.

Utterance System1 requests information about the caller's departure airport, but in User2, the caller takes the initiative to provide information about her destination. In System3, the system's goal is to *implicitly confirm* the destination (because of the possibility of error in the speech recognition component), and *request information* (for the second time) of the caller's departure airport. This combination of communicative goals arises dynamically in the dialog because the system supports user initiative, and requires different capabilities for generation than if the system could only understand the direct answer to the question that it asked in System1.

In User4, the caller provides this information but also takes the initiative to provide the month and day of travel. Given the system's dialog strategy, the communicative goals for its next turn are to *implicitly confirm* all the information that the user has provided so far, i.e. the departure and destination cities and the month and day information, as well as to *request information* about the time of travel. The system's representation of its communicative goals for utterance System5 is in Figure 2. As before, this combination of communicative goals arises in response to the user's initiative.

implicit-confirm(orig-city:NEWARK)
implicit-confirm(dest-city:DALLAS)
implicit-confirm(month:9)
implicit-confirm(day-number:1)
request(depart-time:whatever)

Figure 2: The text plan (communicative goals) for System5 in Dialog D1

Like most working research spoken dialog systems, our current DARPA COMMUNICATOR system uses hand-crafted TEMPLATE-BASED GENERATION. Its output is created by choosing string templates for each elementary speech act, using a large choice function which depends on the type of speech act and various context conditions. Values of template variables (such as origin and destination cities) are instantiated by the dialog manager. The string templates for all the elementary speech acts of a turn are heuristically ordered and then appended, resulting in the system output for that turn. In order to produce output that is not highly redundant, string templates must be written for every possible combination of speech acts in a text plan. Some of the many combinations that can arise are illustrated in Dialog D1. We will refer to the output generated using this approach as the TEMPLATE output. An example TEMPLATE output for the text plan in Figure 2 is in Figure 3.[2]

In the NLG approach, the sentence planner takes as input a text plan such as in Figure 2. In general, the role of the sentence planner is to choose abstract lexico-structural resources for a text plan, where a text plan encodes the communicative goals for an utterance (and, sometimes, their rhetorical structure). The job of the sentence planner is to decide among the large

[2]The different systems in Figure 3 are described below.

System	Realization
Template	Flying from Newark to Dallas, Leaving on the 1st of September, And what time did you want to leave?
Trainable	What time would you like to travel on September the 1st to Dallas from Newark?
RBS (Rule-Based)	What time would you like to travel on September the 1st to Dallas from Newark?
ICF (Rule-Based)	What time would you like to fly on September the 1st to Dallas from Newark?
Random Baseline	Leaving in September. Leaving on the 1st. What time would you, traveling from Newark to Dallas, like to leave?
No Aggregration Baseline	Leaving on the 1. Leaving in September. Going to Dallas. Leaving from Newark. What time would you like to leave?

Figure 3: Sample outputs for each type of generation system used in the evaluation experiment

number of potential realizations of these communicative goals. Some alternative realizations are in Figure 3.

In (Anonymous, 2001) we proposed a new model of sentence planning in which the sentence planner is *automatically trained* from human feedback. We refer to this system as TRAINABLE and describe it in section 3.3. An example TRAINABLE output for the text plan in Figure 2 is in Figure 3. This paper evaluates the relative quality of sentence planning choices that TRAINABLE makes, given a particular text plan input from the dialog manager.

We perform an evaluation using human subjects who judged the TEMPLATE output of our DARPA COMMUNICATOR system, of two different RULE-BASED sentence planners, and of TRAINABLE. In addition, the subjects also judged the output of two BASELINE sentence planners. One baseline, NO AGGREGRATION, simply produces a single sentence for each communicative goal. Another baseline, RANDOM, randomly makes decisions about how to combine communicative goals into sentences.

The experiment required human subjects to read 5 dialogs of real interactions with our DARPA COMMUNICATOR system. At 20 points over the 5 dialogs, the Communicator system's actual utterance (TEMPLATE) is augmented with a set of variants for the system's turn at that point in the dialog. Each set of variants included a representative generated by TRAINABLE, and representatives of the comparison sentence planners described in section 3. At times two or more of these variants coincided, in which case sentences were not repeated and fewer than six sentences were presented to the subjects. The subjects were asked to rate each variation on a 5-point Likert scale, by stating the degree to which they agree with the statement *The system's utterance is easy to understand, well-formed, and appropriate to the dialog context.* A total of 60 subjects, colleagues not involved in research on spoken dialog generation, completed the experiment.

3 Sentence Planning Systems

We evaluate four basic classes of generators: (1) the TRAINABLE sentence planner; (2) the hand-crafted TEMPLATE generator currently operational in our COMMUNICATOR travel planning system; (3) two representative RULE-BASED sentence planners; and (4) two BASELINE sentence planners. This section describes the sentence planners that we compare. The TRAINABLE, RULE-BASED and BASELINE sentence planners are all NLG based sentence planners. In section 3.1, we first describe the shared rule sets of the NLG based sentence planners. Section 3.2 describes the two baselines RANDOM and NO AGGREGRATION. Section 3.3 describes our TRAINABLE sentence planner. Section 3.4 describes the two RULE-BASED sentence planners.

3.1 Aggregation in Sentence Planning

In all of the NLG sentence planners utilized in the evaluation, each speech act is assigned a canonical lexico-structural representation (called a **DSyntS** – Deep Syntactic Structure (Mel'čuk, 1988)). We are thus excluding issues related to lexical choice from this study, and restrict our attention to the question of how elementary structures for separate elementary speech acts are assembled. The basis of all the NLG systems is a set of clause-combining operations that incrementally transform a list of elementary predicate-argument representations (the DSyntSs corresponding to the elementary speech acts of a single text plan) into a list of lexico-structural representations of larger sentences. DSyntSs are combined using the following operations, with examples in Figure 4.

- MERGE. Two identical main matrix verbs can be identified if they have the same arguments; the adjuncts are combined.
- MERGE-GENERAL. Same as MERGE, except that one of the two verbs may be embedded.
- SOFT-MERGE. Same as MERGE, except that the verbs need only to be in a relation of synonymy or hyperonymy (rather than being identical).
- SOFT-MERGE-GENERAL. Same as MERGE-GENERAL, except that the verbs need only to be in a relation of synonymy or hyperonymy.
- CONJUNCTION. This is standard conjunction with conjunction reduction.
- RELATIVE-CLAUSE. This includes participial adjuncts to nouns.
- ADJECTIVE. This transforms a predicative use of an adjective into an adnominal construction.
- PERIOD. Joins two complete clauses with a period.
- RANDOM CUE WORD. Adds a cue word *now, and, alright,* or *okay* to a complete clause.

Rule	Arg 1	Arg 2	Result
MERGE	You are leaving from Newark.	You are leaving at 5	You are leaving at 5 from Newark
MERGE-GENERAL	What time would you like to leave?	You are leaving from Newark.	What time would you like to leave from Newark?
SOFT-MERGE	You are leaving from Newark	You are going to Dallas	You are traveling from Newark to Dallas
SOFT-MERGE-GENERAL	What time would you like to leave?	You are going to Dallas.	What time would you like to fly to Dallas?
CONJUNCTION	You are leaving from Newark.	You are going to Dallas.	You are leaving from Newark and you are going to Dallas.
RELATIVE-CLAUSE	Your flight leaves at 5.	Your flight arrives at 9.	Your flight, which leaves at 5, arrives at 9.
ADJECTIVE	Your flight leaves at 5.	Your flight is nonstop.	Your nonstop flight leaves at 5.
PERIOD	You are leaving from Newark.	You are going to Dallas.	You are leaving from Newark. You are going to Dallas
RANDOM CUEWORD	What time would yo like to leave?	n/a	Now, what time would you like to leave?

Figure 4: List of clause combining operations with examples

These operations are similar to previous aggregation components (Rambow and Korelsky, 1992; Shaw, 1998), although the various MERGE operations are, to our knowledge, novel in this form. Each operation has certain conditions on its applicability, some operations fail when applied to some pairs of DSyntSs.

The result of applying the operations is a **sentence plan tree** (or **sp-tree** for short), which is a binary tree with leaves labeled by all the elementary speech acts from the input text plan, and with its interior nodes labeled with clause-combining operations. The sp-tree is inspired by (Lavoie and Rambow, 1998). The representations used by Danlos (2000), Claire and Webber (1998), or Stone and Doran (1997) are similar, but do not (always) explicitly represent the clause combining operations as labeled nodes. Each node is also associated with a DSyntS: the leaves (which correspond to elementary speech acts from the input text plan) are linked to a canonical DSyntS for that speech act (by lookup in a hand-crafted dictionary). The interior nodes are associated with DSyntSs which result from executing the clause-combing operation with which the node is labeled on its two daughter nodes. (A PERIOD node results in a DSyntS headed by a period and whose daughters are the two daughter DSyntSs – see Danlos (2000) for the use of the period as a node label in lexico-syntactic representations.) As a result, the DSyntS for the entire turn is associated with the root node. This DSyntS is sent to the surface realizer, which returns a sentence (or several sentences, if the DSyntS contains period nodes). We utilize the RealPro surface realizer with all the sentence planners described below (Lavoie and Rambow, 1998).

As an example Figure 5 shows the sp-tree for utterance System5 in Dialog D1. Node **soft-merge-general**$_3$ merges an implicit-confirmation of the destination city and the origin city. The row labelled SOFT-MERGE in Figure 4 shows the result of applying the soft-merge operation when Args 1 and 2 are implicit confirmations of the origin and destination cities.

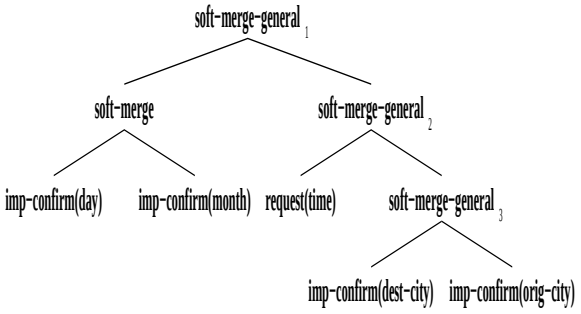

Figure 5: A Sentence Plan Tree for Utterance System 5 in Dialog D1

The problem of sentence planning for the purpose of this study is the problem of how to construct the sentence plan tree. The different sentence planners that we evaluate and describe below differ only in how the sp-tree is constructed.

3.2 Baseline Sentence Planners

One obvious possible baseline system results from constructing the sp-tree by applying only the PERIOD operation: each elementary speech act is realized as its own sentence. This baseline was suggested by Hovy and Wanner (1996). This provides one possible realization of a turn, which we refer to as NO AGGREGATION. Figure 3 includes a NO AGGREGATION output for the text plan in Figure 2.

A second possible baseline sentence planner simply applies combination rules randomly according to a hand-crafted probability distribution based on preferences for operations such as the MERGE family over

CONJUNCTION and PERIOD. In order to be able to generate the resulting sentence plan tree, we exclude certain combinations, such as generating anything other than a PERIOD above a node labeled PERIOD in a sentence plan. The resulting realization we refer to as RANDOM. Figure 3 includes a RANDOM output for the text plan in Figure 2.

In order to construct a more complex, and hopefully better, sentence planner, we need to encode constraints on the application of, and ordering of, the operations. It is here that the remaining approaches differ. In the first approach, TRAINABLE, we learn constraints from training material; in the second approach, **rule-based**, we construct constraints by hand.

3.3 A Trainable Sentence Planner

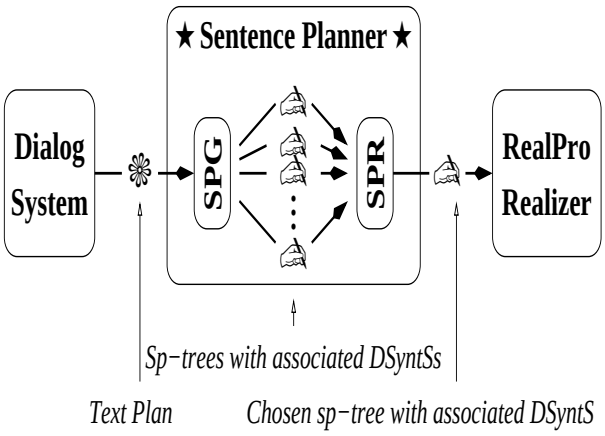

Figure 6: Architecture of the Trainable Sentence Planner

For the sentence planner TRAINABLE, we reconceptualize sentence planning as consisting of two distinct phases as shown in in Figure 6. In the first phase, the sentence-plan-generator (**SPG**) randomly generates up to twenty possible sentence plans for a given text-plan input. For this phase we use the RANDOM sentence-planner as described above, with a probabilistic bias to prefer rules according to heuristic preferences. This allows us to bias the random generation towards plans that are more likely to be high quality, while generating a relatively smaller sample of sentence plans. In the second phase, the sentence-plan-ranker (**SPR**) ranks the sample sentence plans, and then selects the top-ranked output to input to the surface realizer. We used RankBoost (Freund et al., 1998) to automatically learn rules from training data, in a spirit similar to that of (Collins, 2000). The training data was assembled by using RANDOM to randomly generate up to 20 realizations for 100 turns; two human judges then ranked each of these realizations (using the setup described in Section 2). Over 3,000 features were discovered from the generated trees by routines that explore and encode various structural and lexical aspects of the sp-tree and the DSyntS for that realization. Each feature exists in both a general version as well as a lexicalized version. RankBoost then identified a subset of features which contribute most to a realization's ranking. The **SPR** uses these rules to rank alternative sp-trees (and the associated DSyntSs), and then selects the top-ranked output as input to the surface realizer. Anonymous (2001) describes TRAINABLE in detail.

3.4 Two Rule-Based Sentence Planners

It has not been the object of our research to construct a rule-based sentence planner, be it domain-independent or optimized for our domain. Our goal was to compare the TRAINABLE sentence planner with a representative rule-based system. We decided against using an existing off-the-shelf rule-based system, since it would be too complex a task to port it to our application. Instead, we constructed two rule-based sentence planners which could serve as reasonable representatives for comparison with TRAINABLE. This task was made easier by the fact that we could reuse much of the work done for TRAINABLE, in particular the data structure of the sp-tree and the implementation of the clause-combining operations. We developed the systems by applying heuristics for producing good output, such as preferences for aggregation. However there were no guidelines for ordering the combinations of speech acts that we see in the text plans for spoken dialog systems. Since it was not clear which ordering would be optimal across all text plans, we constructed two rule-based systems that differed only in the initial ordering of the communicative acts to be combined into an sp-tree.

In the first rule-based system, which we call RULE-BASED SYSTEM or RBS for short, we first order the elementary speech acts so that explicit confirms come first, followed by requests, and finally followed by implicit confirms. Note that explicit confirms and requests do not co-occur in our data set. In the initial step, we take the two leftmost members of the text plan and try to combine them using one of the combination operations, which are preference-ranked in the following order (from most preferred to least preferred): ADJECTIVE, MERGE, CONJUNCTION, RELATIVE-CLAUSE, PERIOD. The first operation to succeed is chosen. This yields a binary splan-tree with three nodes, which becomes the **current splan-tree**. As long as the root node of the current splan-tree is not a PERIOD, we iterate through the list of remaining elementary speech acts on the ordered text plan, combining each one with the current splan-tree using the preference-ranked operations as just described. The

result of each iteration step is a binary, left-branching splan-tree. However, if the root node of the current splan-tree is a PERIOD, we start a new current splan-tree, as in the initial step described above. When the text plan has been exhausted, all partial splan-trees (all except for the last one which is rooted in PERIOD) are combined in a left-branching tree using PERIOD. We then add cue words according to the following rules: (1) The cue word *now* is attached to utterances beginning a new subtask; (2) The cue word *and* is attached to utterances continuing a subtask; (3) The cue words *alright* or *okay* are attached to utterances containing implicit confirmations.

The second second rule-based system is identical to the first, except that the text plan is ordered so that implicit confirms come first rather than last. This system we call **Rule-based System with Implicit Confirms First** or ICF for short. Figure 3 includes an RBS and an ICF output for the text plan in Figure 2. In this case ICF and RBS differ only in the verb chosen as a more general verb during the SOFT-MERGE operation.

We illustrate the RBS procedure using an example; ICF works similarly for this example. For RBS, the text plan in Figure 2 is ordered so that the request speech act is first. For the request, a DSyntS is chosen that can be paraphrased as *What time would you like to leave?*. Then, the first implicit-confirm is translated by lookup into a DSyntS which on its own could generate *Leaving in September*. We first try the ADJECTIVE aggregation operation, but since neither tree is a predicative adjective, this fails. We then try the MERGE family. MERGE itself fails since the root node of the first DSyntS is not labeled *travel* as is the root node of the second (it is labeled *like*), but MERGE-GENERAL succeeds, since the tree for the request has an embedded node labeled *leave*. The resulting DSyntS can be paraphrased as *What time would you like to leave in September?*, and is attached to the new root node of the resulting sp-tree. The root node is labeled "merge-general", and its two daughters are the two speech acts. The implicit-confirm of the day is added in a similar manner (adding another left-branching node to the sp-tree), yielding a DSyntS that can be paraphrased as *What time would you like to leave on September the 1st?* (using some special-case attachment for dates within MERGE). We now try and add the DSyntS for the implicit-confirm, whose DSyntS might generate *Going to Dallas.* Here, we again cannot use ADJECTIVE, but we cannot use MERGE or MERGE-GENERAL, since no merge matches. Instead, we use SOFT-MERGE-GENERAL, which allows us to identify the *leave* node with the *go* root node of the DSyntS of the implicit-confirm. When soft-merging *leave* with *go*, *fly* is chosen as a generalization, resulting in a DSyntS that can be generated as *What time would you like to fly on September the 1st to Dallas?*. The sp-tree has added a layer but is still left-branching. Finally, the last implicit-confirm is added to yield a DSyntS that is realized as *What time would you like to fly on September the 1st to Dallas from Newark?*.

4 Experimental Results

System	Min	Max	Mean	S.D.
Template	1	5	3.9	1.1
Trainable	1	5	3.9	1.3
RBS	1	5	3.4	1.4
ICF	1	5	3.5	1.4
No Aggregation	1	5	3.0	1.2
Random	1	5	2.7	1.4

Figure 7: Summary of Overall Results for all Systems Evaluated

All 60 subjects typically completed the experiment in a half hour or less. The experiment resulted in a total of 1200 judgements for each of the systems being compared, since each subject judged 20 utterances by each system. We first discuss overall differences among the different systems and then make comparisons among the four different types of systems: (1) TEMPLATE, (2) TRAINABLE, (3) **rule-based** and (4) **baseline**. All statistically significant results discussed here had p values of less than .01.

We first examined whether differences in human ratings (score) was predictable from the type of system that produced the utterance being rated. A one-way ANOVA with SYSTEM as the independent variable and SCORE as the dependent variable showed that there were significant differences in SCORE as a function of SYSTEM. The overall differences are summarized in Figure 7.

As Figure 7 indicates, some system outputs received more consistent scores than others, e.g. the standard deviation for TEMPLATE was much smaller than RANDOM. The Figure also shows that the ranking of the systems by average score is **template**, TRAINABLE, ICF, RBS, NO AGGREGATION, and RANDOM. Posthoc comparisons of the scores of individual pairs of systems using the adjusted Bonferroni statistic revealed several different groupings.[3]

The highest ranking systems were TEMPLATE and TRAINABLE, whose human ratings were not statistically significantly different from one another. This shows that it is possible to match the quality of a handcrafted system with a trainable one, which should be

[3]The adjusted Bonferroni statistic guards against accidentally finding differences between systems when making multiple comparisons among systems.

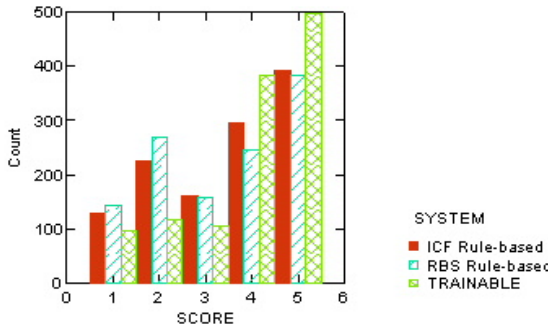

Figure 8: Bar chart comparing distribution of human ratings for TRAINABLE and the **rule-based** sentence planners ICF and RBS

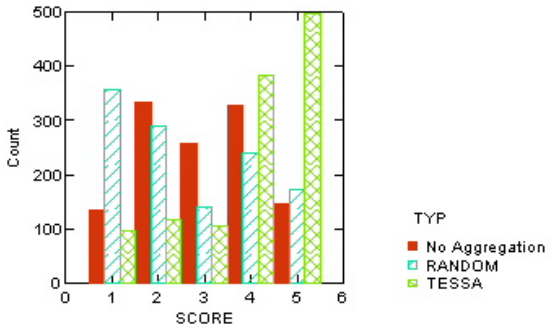

Figure 9: Bar chart comparing distribution of human ratings for TRAINABLE and the BASELINE sentence planners NO AGGREGATION and RANDOM.

more portable, more general and require less overall engineering effort.

The next group of systems were the two rule-based systems, ICF and RBS, which were not statistically different from one another. However TRAINABLE was statistically better than both of these systems ($p < .01$). Figure 4 shows the distribution of human scores for ICF, RBS and TRAINABLE. The Figure shows that while all the system rankings are skewed towards higher rankings, TRAINABLE got many more high rankings than either of the rule based systems. In a sense this may not be that surprising, because as Hovy and Wanner (1996) point out, it is difficult to construct a rule-based sentence planner that handles all the rule interactions in a reasonable way. Both the general and the lexicalized features that the trainable sentence planner's **SPR** uses allow TRAINABLE to be sensitive to particular discourse configurations or lexical collocations. In order to encode these in a rule-based sentence planner, one would first have to discover these constraints and then determine a way of enforcing them. However the **SPR** simply learns that a particular configuration is less preferred, resulting in a small decrement in ranking for the corresponding sp-tree. This flexibility of incrementing or decrementing a particular sp-tree by a small amount may in the end allow it to be more sensitive to small distinctions than a rule-based system.

Along with the TEMPLATE and RULE-BASED systems, TRAINABLE also scored better than the baseline systems NO AGGREGATION and RANDOM. This is also somewhat to be expected, since the baseline systems were intended to be the simplest systems constructable. However it would have been a possible outcome for TRAINABLE to not be different than either one of these systems; for example if the sp-trees produced by RANDOM were all equally good, or if the aggregation rules that TRAINABLE learned produced output less readable than the NO AGGREGATION baseline. Figure 4 shows the distribution of scores for NO AGGREGATION, RANDOM and TRAINABLE. The Figure shows that the distributions of scores for TRAINABLE vs. the baseline systems were very different, with TRAINABLE skewed towards higher scores, **Random** skewed towards lower scores and the NO AGGREGATION distribution approximately normal.

Interestingly NO AGGREGATION also scored significantly better than RANDOM ($p < .01$). The standard deviation of the scores for NO AGGREGATION was also smaller than RANDOM (see Figure 7). Remember that RANDOM produced random sp-trees, which often resulted in an arbitrary ordering of the speech acts in the output. While NO AGGREGATION produced fully redundant utterances with no aggregation, NO AGGREGATION output placed the initiative taking speech act at the end of the utterance in its most natural position. This may have been a significant factor in the preference for NO AGGREGATION over RANDOM. Other source of the preference for NO AGGREGATION could have been its predictability.

5 Discussion and Future Work

Previous work on evaluation of natural language generation has utilized three different approaches to evaluation. The first approach is a subjective evaluaton methodology such as we use here, where human subjects rate NLG outputs produced by different sources (Lester and Porter, 1997). Other work has evaluated template-based spoken dialog generation with a task-based approach, i.e. the choices that the generator makes are evaluated with a task metric such as task completion or user satisfaction after dialog completion (Walker et al., 1998). This approach can work well when the task only involves one or two exchanges, when the choices have large effects over the whole dialog, or the choices vary the content of the utterance. Because sentence planning choices realize the same content and only affect the current utterance, we believed it important to get local feedback. A final approach focuses on subproblems of natural language generation such as the generation of referring expressions. For this type of problem it is possible to evaluate the generator by the degree to which it matches

human performance(Yeh and Mellish, 1997). When evaluating sentence planning, this approach doesn't make sense. One of the difficult things about evaluating a sentence planner is that many different realizations may be equally good.

This paper presented an exhaustive evaluation of our trainable sentence planner TRAINABLE in which we compared it to the TEMPLATE output of our working COMMUNICATOR system, to several RULE-BASED sentence planners, and to several BASELINE sentence planners. The results validate our methodology for developing a trainable sentence planner. The results show that TRAINABLE outperforms two representative rule-based sentence planners, and that TRAINABLE performs as well as the hand-crafted TEMPLATE system, but is more easily and quickly tuned to a new domain: the training materials for the TRAINABLE sentence planner can be collected from subjective judgements from users with little or no linguistic knowledge.

However this experiment did not show that trainable sentence planners produce, in general, better-quality output than template-based or rule-based sentence planners. That would be impossible: given the nature of these systems, any quality standard that may have been set for the output can be met given sufficient person-hours, elapsed time, and software engineering acumen. Our principal goal, rather, is to show that the quality of the TEMPLATE output, for a currently operational dialog system whose template-based output component was developed, expanded, and refined over about 18 months, can be achieved using a trainable system, for which the necessary training data was collected in three person-days. Furthermore, we wished to show that a representative rule-based system based on current literature, without massive domain-tuning, cannot achieve the same level of quality.

In future work, we hope to expand the capabilities of the trainable sentence planner and integrate it into our COMMUNICATOR system.

References

Anonymous. 2001. A trainable sentence planner for spoken dialogue systems. In *Proceedings of the North American Meeting of the Association for Computational Linguistics*.

Gardent Claire and Bonnie Webber. 1998. Varieties of ambiguity in incremental discourse processing. In *Proceedings of AMLap-98 (Architectures and Mechanisms for Language Processing)*, Freiburg, Germany.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Laurence Danlos. 2000. G-TAG: A lexicalized formalism for text generation inspired by tree adjoining grammar. In Anne Abeillé and Owen Rambow, editors, *Tree Adjoining Grammars: Formalisms, Linguistic Analysis, and Processing*. CSLI Publications.

Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. 1998. An efficient boosting algorithm for combining preferences. In *Machine Learning: Proceedings of the Fifteenth International Conference*. Extended version available from http://www.research.att.com/ schapire.

E.H. Hovy and Leo Wanner. 1996. Managing sentence planning requirements. In *Proceedings of the ECAI'96 Workshop Gaps and Bridges: New Directions in Planning and Natural Language Generation*.

Benoit Lavoie and Owen Rambow. 1998. A framework for customizable generation of multi-modal presentations. In *36th Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL'98)*, Montréal, Canada. ACL.

James Lester and Bruce Porter. 1997. Developing and empirically evaluating robust explanation generators: The knight experiments. *Computational Linguistics*, 23-1:65–103.

Igor A. Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. State University of New York Press, New York.

Owen Rambow and Tanya Korelsky. 1992. Applied text generation. In *Proceedings of the Third Conference on Applied Natural Language Processing, ANLP92*, pages 40–47.

James Shaw. 1998. Clause aggregation using linguistic knowledge. In *Proceedings of the 8th International Workshop on Natural Language Generation*, Niagara-on-the-Lake, Ontario.

Matthew Stone and Christine Doran. 1997. Sentence planning as description using tree adjoining grammar. In *35th Meeting of the Association for Computational Linguistics (ACL'97)*, pages 198–205, Madrid, Spain.

M. Walker, J. Fromer, G. Di Fabbrizio, C. Mestel, and D. Hindle. 1998. What can I say: Evaluating a spoken language interface to email. In *Proceedings of the Conference on Computer Human Interaction (CHI 98)*, pages 582–589.

Ching-Long Yeh and Chris Mellish. 1997. An empirical study on the generation of anaphora in chinese. *Computational Linguistics*, 23-1:169–190.