Article Submitted to Computer Speech and Language

Training a Sentence Planner for Spoken Dialogue Using Boosting

Marilyn A. Walker[1], Owen C. Rambow[2] and Monica Rogati[3]

[1] *AT&T Labs – Research*
Florham Park, NJ, USA
walker@research.att.com

[2] *University of Pennsylvania*
Philadelphia, PA, USA
rambow@unagi.cis.upenn.edu

[3] *Carnegie Mellon University*
Pittsburgh, PA, USA
mrogati+@cs.cmu.edu

Abstract

In the past few years, as the number of dialogue systems has increased, there has been an increasing interest in the use of natural language generation in spoken dialogue. Our research assumes that trainable natural language generation is needed to support more flexible and customized dialogues with human users. This paper focuses on methods for automatically training the sentence planning module of a spoken language generator. Sentence planning is a set of inter-related but distinct tasks, one of which is sentence scoping, i.e. the choice of syntactic structure for elementary speech acts and the decision of how to combine them into one or more sentences. The paper first presents SPoT, a trainable sentence planner, and a new methodology for automatically training SPoT on the basis of feedback provided by human judges. Our methodology is unique in neither depending on hand-crafted rules nor on the existence of a domain-specific corpus. SPoT first randomly generates a candidate set of sentence plans and then selects one. We show that SPoT learns to select a sentence plan whose rating on average is only 5% worse than the top human-ranked sentence plan. We then experimentally evaluate SPoT by asking human judges to compare SPoT's output with a hand-crafted template-based generation component, two rule-based sentence planners, and two baseline sentence planners. We show that SPoT performs better than the rule-based systems and the baselines, and as well as the hand-crafted system.

1. Natural Language Generation in Dialogue Systems

The past several years have seen a large increase in commercial spoken dialogue systems. These systems typically utilize system-initiative dialogue 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 [3]. Second, there is a perceived need for spoken dialogue systems to be more flexible and support user initiative, but this also requires greater flexibility for system utterance generation. Finally, dialogue 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 dialogue 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, there is abundant anecdotal evidence that maintenance of the collection of templates becomes a software engineering problem as the complexity of the dialogue system increases.

The second approach is NATURAL LANGUAGE GENERATION (NLG), which customarily divides the generation process into three modules [24, 36]:

- During **text planning**, a high-level communicative goal is broken down into a structured representation of atomic communicative goals, i.e., goals that can be attained with a single communicative act (in language, by uttering a single clause). The atomic communicative goals may be linked by rhetorical relations which show how attaining the atomic goals contributes to attain the high-level goal.
- During **sentence planning**, abstract linguistic resources are chosen to achieve the atomic communicative goals. This includes choosing meaning-bearing lexemes, and how the meaning-bearing lexemes are connected through abstract grammatical constructions. As a side-effect, sentence planning also determines sentence boundaries: this happens when syntactic means are chosen which combine linguistic resources for several communicative goals

into one sentence (a process known as *aggregation*). There need not be, and usually is not, a one-to-one relation between elementary communicative goals and sentences in the final text.

- During **realization**, the abstract linguistic resources chosen during sentence planning are transformed into a surface linguistic utterance by adding function words (such as auxiliaries and determiners), inflecting words, and determining word order. This phase is not a planning phase in that it only executes decisions made previously.

The proposed architecture for a spoken dialogue system that incorporates NLG is shown in Figure 1. NLG promises improved system output by allowing the Prosody Assignment component to have access to all of the previous levels of representation. NLG also promises portability across application domains and dialogue 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.

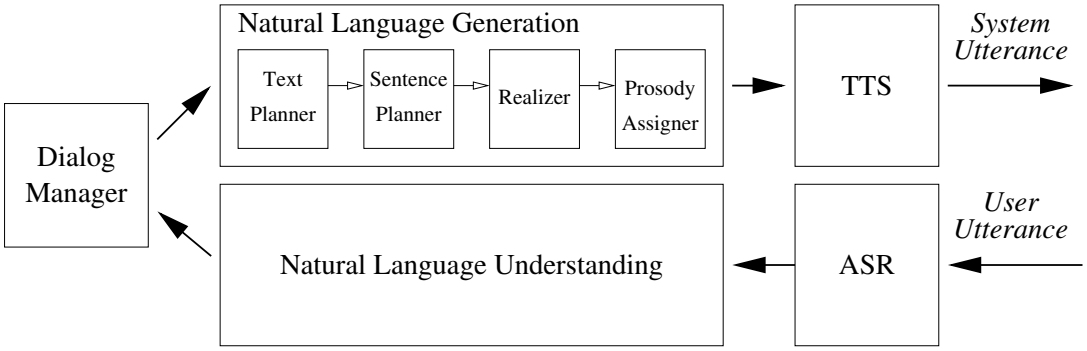

Figure 1: Architecture of a dialogue system with natural language generation

Recently, several different techniques for *automatically training* different modules of an NLG system have been proposed [20, 29, 45, 1]. 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. The research reported in this article concerns developing a trainable SENTENCE PLANNING module for AT&T's mixed-initiative DARPA Communicator system for travel planning, AMELIA [26, 44], In AMELIA, the sentence planner must be able to generate sentence plans for a large number of combinations of communicative goals arising in many different contexts.

In this article, we propose a new model of sentence planning called SPoT. In

SPoT, the sentence planner is automatically trained, using feedback from two human judges, to choose the best from among different options for realizing a set of communicative goals.

We evaluate the performance of the learning component of SPoT, and show that SPoT learns to select sentence plans whose rating on average is only 5% worse than the top human-ranked sentence plan. While this evaluation shows that SPoT has indeed learned from the human judges, it does not show that using only two human judgments is sufficient to produce more broadly acceptable results, nor does it show that SPoT performs as well as optimized hand-crafted template or rule-based systems. In order to explore this issue, we conducted a second set of experiments to evaluate SPoT. Because SPoT is trained on data from AT&Ts DARPA Communicator system AMELIA, we can directly compare SPoT to the hand-crafted, template-based generation component of AMELIA. In order to perform an extensive 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 these different approaches in an evaluation experiment in which 60 human subjects were asked to compare the quality of each system's output by rating it on a scale of 1 to 5. We show that SPoT performs better than both rule-based systems and as well as AMELIA's hand-crafted template-based system. These four systems outperform the baseline sentence planners.

In the remainder of the paper, Section 2 describes the sentence planning task in more detail. We then describe the sentence plan generator (**SPG**) in Section 3, the sentence plan ranker (**SPR**) in Section 4, and the results of training in Section 5. The evaluation experiment is described in Section 6. The sentence planners used in the evaluation are described in Section 6.1 and Section 6.2. Section 6.3 then presents the evaluation results. We delay the discussion of related work to Section 7 when we can compare it with our approach. Section 8 summarizes our results and discusses future work.

2. The Sentence Planning Task

The term "sentence planning" comprises many distinct tasks and many ways of organizing these tasks have been proposed in the literature. In general, the role of the sentence planner is to choose abstract linguistic resources (meaning-bearing lexemes, syntactic constructions) for a text plan. For example, consider the required capabilities of a sentence planner for AMELIA as illustrated in Dialogue D1.

(D1) 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: What time would you like to travel on September the 1st to Dallas from Newark?

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, AMELIA'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. In User4, the caller provides this information but also provides the month and day of travel. Given AMELIA's current dialogue 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 that has not yet been provided by the user. The system's representation of its communicative goals for utterance System5 is in Figure 2. The job of the sentence planner is to decide among the large number of potential realizations of these communicative goals. Some example alternative realizations are in Figure 3. The meaning of the human ratings and RankBoost scores in Figure 3 are discussed below.

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

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

Alt	Realization	H	RB
0	What time would you like to travel on September the 1st to Dallas from Newark?	5	.85
5	Leaving on September the 1st. What time would you like to travel from Newark to Dallas?	4.5	.82
13	Now, what time would you like to leave? Flying to DALLAS from Newark on September the 1.	3	.58
8	Leaving in September. Leaving on the 1st. What time would you, traveling from Newark to Dallas, like to leave?	2	.39

Figure 3: Alternative sentence plan realizations for the text plan for utterance System5 in dialogue D1. H = human rating, RB = RankBoost score.

In order to train SPoT, we reconceptualize the task of the sentence planner as consisting of two distinct phases. In the first phase, the sentence-plan-generator (**SPG**) *generates* a potentially large sample of possible sentence plans for a given text-plan input. In the experiments reported below the sentence-plan-generator (**SPG**) generates 12-20 possible sentence plans for a given input text plan. Each speech act in the text plan is assigned a canonical lexico-structural representation (called a **DSyntS** – Deep Syntactic Structure [30]). The sentence plan is a tree recording how these elementary DSyntS are combined into larger DSyntSs; the DSyntS for the entire input text plan is associated with the root node of the tree. In the second phase, the sentence-plan-ranker (**SPR**) *ranks* the sample sentence plans generated by the **SPG**, and then selects the top-ranked output to input to the surface realizer RealPro [21]. It would also be consistent with our model for SPoT to pass a list of N-best sentence plans along with their rankings to the surface realizer and prosodic assignment components [4]. Given one or more sentence-plans to rank, the **SPR** uses rules automatically learned from training data, using techniques similar to [8, 12]. The method we propose for training a sentence planner is unique in neither depending on hand-crafted rules, nor on the existence of a text or speech corpus in the domain of the sentence planner obtained from the interaction of a human with a system or another human. The architecture is summarized in Figure 4.

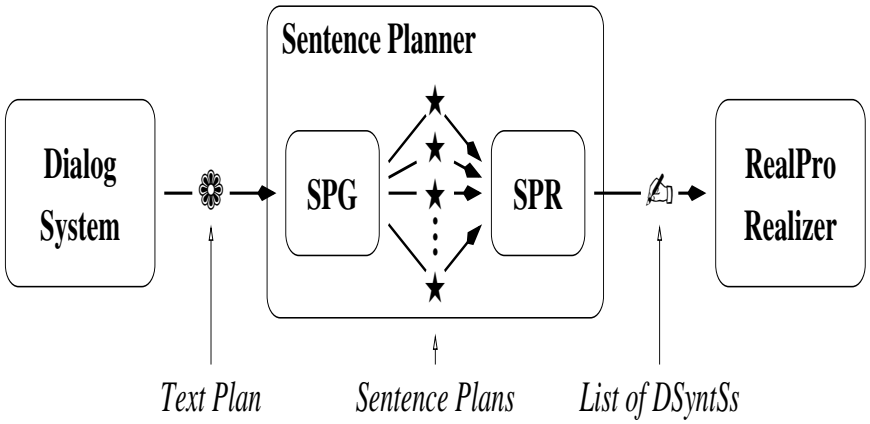

Figure 4: Architecture of SPoT

3. The Sentence Plan Generator

The research presented here is primarily concerned with creating a trainable **SPR**. A strength of our approach is the ability to use a very simple **SPG**, as we explain below. The basis of our **SPG** is a set of clause-combining operations that incrementally transform a list of elementary predicate-argument representations (the DSyntSs corresponding to elementary speech acts, in our case) into a single

Rule	Sample first argument	Sample second argument	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

Figure 5: List of clause combining operations with examples from our domain; an explanation of the operations is given in Section 3.

lexico-structural representation, by combining these representations using the following combining operations. Examples can be found in Figure 5. Since the combination of clauses depends to a certain extent on language-specific syntax, the "semantic" representations must already be quite close to a lexical-structural representation, and we use a lexical predicate-argument structure, namely the "Deep-Syntactic Structure" (DSyntS) of Meaning-Text Theory [30].

- 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.

These operations are not domain-specific and are similar to those of previous aggregation components [36, 40, 9], although the various MERGE operations are, to our knowledge, novel in this form. In addition, we use a rule which does not combine two clauses but instead modifies a single clause:

- RANDOM-CUEWORD adds a cueword randomly from among *All right*, *now*, *OK*, or *and*.

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 [22]. The representations used by Danlos, Gardent and Webber, and Stone and Doran [9, 13, 42] are similar, but do not (always) explicitly represent the clause combining operations as labeled nodes. In our representation of the sp-tree, 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). It would also be consistent with our approach for a **set** of DSyntSs to be associated with each speech act, rather than a single DSyntS as in these experiments.

The interior nodes are associated with DSyntSs by executing their clause-combining operation on their two daughter nodes. (A PERIOD node results in a DSyntS headed by a period and whose daughters are the two daughter DSyntSs.) If a clause combination fails, the sp-tree is discarded (for example, if we try to create a relative clause of a structure which already contains a period). As a result, the DSyntS for the entire turn is associated with the root node. This DSyntS can be sent to RealPro, which returns a sentence (or several sentences, if the DSyntS contains period nodes). The **SPG** is designed in such a way that if a DSyntS is associated with the root node, it is a valid structure which can be realized.

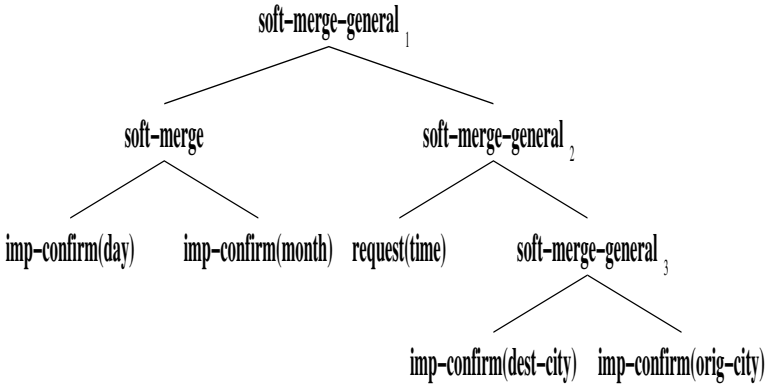

Figure 6: Alternative 0 Sentence Plan Tree

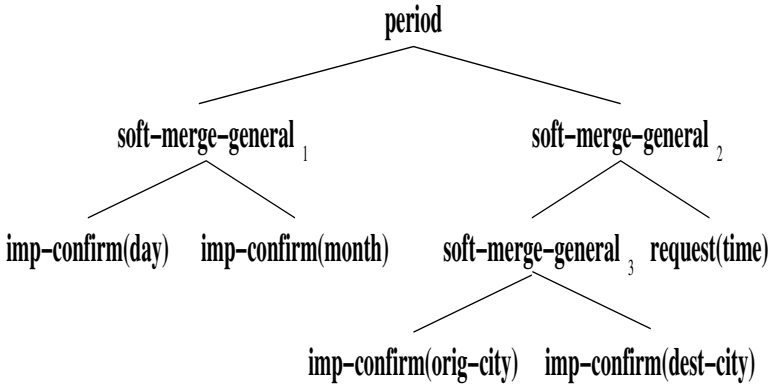

Figure 7: Alternative 5 Sentence Plan Tree

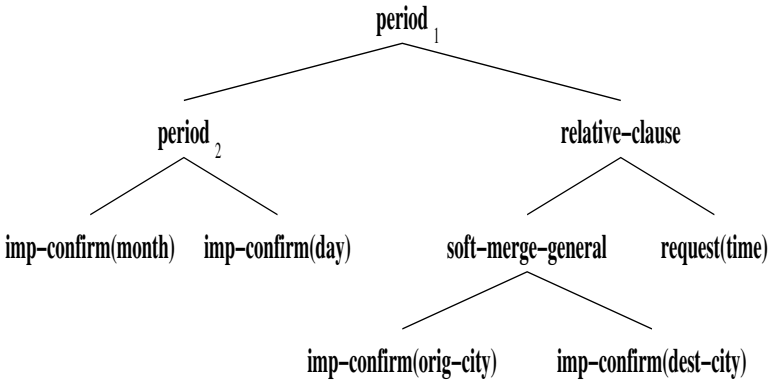

Figure 8: Alternative 8 Sentence Plan Tree

Figure 3 shows some of the realizations of alternative sentence plans generated by our **SPG** for utterance System5 in Dialogue D1. Sp-trees for alternatives 0, 5, and 8 are in Figures 6, 7, and 8. For example, consider the sp-tree in Figure 8. Node **soft-merge-general** merges an implicit-confirmation of the destination city and the origin city. The row labelled SOFT-MERGE in Figure 5 shows the result of applying the soft-merge operation when Args 1 and 2 are implicit confirmations of the origin and destination cities. Figure 9 illustrates the relationship between the sp-tree and the DSyntS for alternative 8. The labels and arrows show the DSyntSs associated with each node in the sp-tree (in Figure 8), and the diagram also shows how structures are composed into larger structures by the clause combining operations.

The complexity of most sentence planners arises from the attempt to encode constraints on the application of, and ordering of, the operations, in order to generate a single high quality sentence plan. In our approach, we do not need to encode such constraints. Rather, we generate a random sample of possible sentence plans for each text plan, up to a pre-specified maximum number of

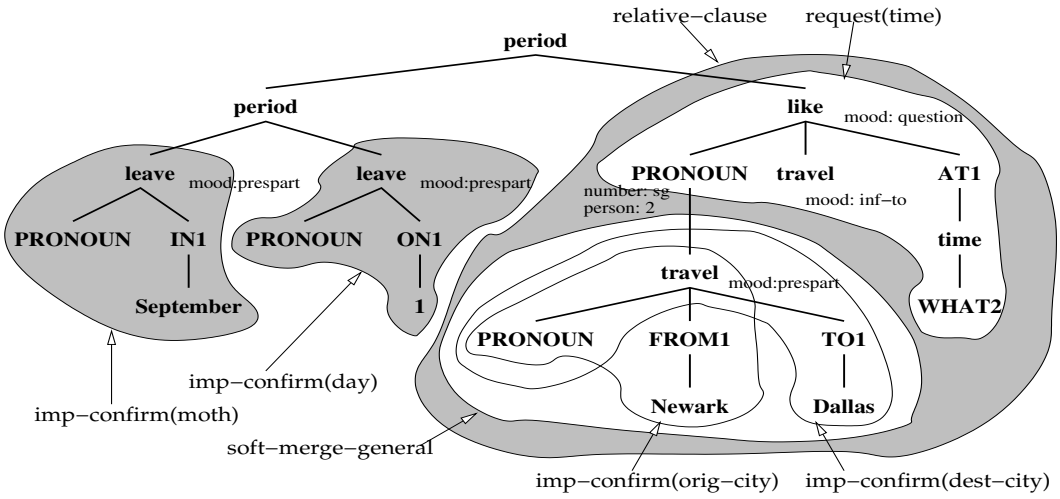

Figure 9: Alternative 8 DSyntS (not all linguistic features are shown)

sentence plans, by randomly selecting among the operations according to some probability distribution. Here the probability distribution is hand-crafted based on assumed preferences for operations such as SOFT-MERGE and SOFT-MERGE-GENERAL over CONJUNCTION and PERIOD. This allows us to bias the **SPG** to generate plans that are more likely to be high quality, while generating a relatively smaller sample of sentence plans. We could also train in two phases where the goal of the first phase would be to learn this probability distribution.

4. The Sentence-Plan-Ranker

The sentence-plan-ranker **SPR** takes as input a set of sentence plans generated by the **SPG** and ranks them. In order to train the **SPR** we applied the machine learning program RankBoost [12], to learn from a labelled set of sentence-plan training examples a set of rules for scoring sentence plans. Our motivation for treating sentence planning as a ranking problem is that it seems clear that, for many generation problems, there is no single correct answer, but rather a partial order of acceptability for many different solutions. In the remainder of the section, we describe how we train the **SPR**. Section 4.1 first describes the training data and how it was collected. Section 4.2 presents the RankBoost algorithm. Section 4.3 describes the process for automatically generating the feature sets used as input to RankBoost and the features themselves.

4.1. Examples and Feedback

To apply RankBoost, we require a set of example sp-trees, each of which have been rated, and encoded in terms of a set of features (see below). We started with a corpus of 100 text plans generated in context in 25 dialogues by the AMELIA dialogue system. We modified the dialogue manager of AMELIA to generate text

plans by writing out a text-plan to the logfile for each communicative goal that AMELIA achieved by sending a template to the TTS engine. We then extracted the text plans that were generated in context from the logfiles to produce a set of 100 text plans. We then ran the **SPG**, parameterized to generate at most 20 distinct sp-trees for each text plan. Since not all text plans have 20 valid sp-trees (while some have many more), this resulted in a corpus of 1868 sentence plans. These 1868 sp-trees, realized by RealPro, were then rated by two judges (the first two authors of this paper), who are native speakers, in the context of the transcribed original dialogues (and therefore also with respect to their adequacy given the communicative goals for that turn). The judges were asked to indicate their degree of agreement with the statement: *The system's utterance is easy to understand, well-formed, and appropriate to the dialogue context* on a scale from 1 to 5. The ratings given by the judges were then averaged to provide a rating between 1 and 5 for each sentence plan alternative. Figure 10 shows that the rankings assigned to the sentence plans were normally distributed with a range from 1 to 5; the mean was 2.86 and the median was 3. Approximately 97% of the text plans had at least one sentence-plan output that was ranked 4 or higher. This, along with the normal distribution, indicates that the sentence plan generator had the capability to generate high quality sentence plans, but that it is not trivial to do so by random selection of operations. Each sp-tree provided an example input to RankBoost, and each corresponding rating was the feedback for that example.

Figure 10: Human rankings for all 1868 sentence plans

4.2. RankBoost

RankBoost is a member of a family of boosting algorithms [39]. The boosting algorithm for ranking is described in detail elsewhere [12]: for completeness, we give a brief description in this section. To train the **SPR** each example x is represented by a set of m indicator functions $h_s(x)$ for $1 \leq s \leq m$. The indicator

functions are calculated by thresholding the feature values (counts) described in Section 4.3. For example, one such indicator function might be

$$h_{100}(x) = \begin{cases} 1 & \text{if DSYNT-TRAVERSAL-PRONOUN}(x) \\ & \geq 2 \\ 0 & \text{otherwise} \end{cases} .$$

So $h_{100}(x) = 1$ if the number of pronouns in x is ≥ 2. A single parameter α_s is associated with each indicator function, and the "ranking score" for an example x is then calculated as

$$F(x) = \sum_s \alpha_s h_s(x)$$

This score is used to rank competing sp-trees of the same text plan in order of their 'goodness'. The training examples are used to set the parameter values α_s. The human judgments are converted into a training set of *ordered pairs* of examples x, y, where x and y are candidates for the same sentence, and x is strictly preferred to y. More formally, the training set $\mathcal{T}$ is

$$\mathcal{T} = \{(x,y) |\ x, y \text{ are realizations for the same text plan,} \\ x \text{ is preferred to } y \text{ by human judgments}\}$$

Thus each text plan with 20 candidates could contribute up to $(20 * 19)/2 = 190$ such pairs: in practice, fewer pairs could be contributed due to different candidates getting tied scores from the annotators.

Training is the process of setting the parameters α_s to minimize the following loss function:

$$Loss = \sum_{(x,y)\in\mathcal{T}} e^{-(F(x)-F(y))}$$

It can be seen that as this loss function is minimized, the values for $(F(x)-F(y))$ where x is preferred to y will be pushed to be positive, so that the number of ranking errors (cases where ranking scores disagree with human judgments) will tend to be reduced. Initially all parameter values are set to zero. The optimization method then greedily picks a single parameter at a time – the parameter which will make the most impact on the loss function – and updates the parameter value to minimize the loss. The result is that substantial progress is typically made in minimizing the error rate, with relatively few non-zero parameter values. Freund *et. al* show that under certain conditions the combination of minimizing the loss function while using relatively few parameters leads to good generalization on test data examples [39]. Empirical results for boosting (including ours) have shown that in practice the method is highly effective [12].

4.3. Features Used by RankBoost

Rankboost, like other machine learning programs of the boosting family, can handle a very large number of features. Therefore, instead of carefully choosing

a small number of features by hand which may be useful, we generated a very large number of features and let RankBoost choose the relevant ones. In total, we used 3,291 features in training the **SPR**. Features were discovered from the actual sentence plan trees that the **SPG** generated through the feature derivation process described below, in a manner similar to that used by [8]. The motivation for the features was to represent in a declarative way control decisions that were taken by the randomized **SPG** during the construction of the sp-tree. To encourage the learning of general rules, we avoided features specific to particular text plans by discarding those that occurred fewer than 10 times.

Features are derived from two sources: the sp-trees and the DSyntSs associated with the root nodes of sp-trees. The feature names are prefixed with "sp-" or "dsynt-" depending on the source. There are two types of features: local and global. Local features record structural configurations local to a particular node, i.e., that can be described with respect to a single node (such as its ancestors, its daughters, etc.). The value of the feature is the number of times this configuration is found in the sp-tree or DSyntS. Each type of local feature also has a corresponding parameterized or lexicalized version, which is more specific to aspects of the particular dialogue in which the text plan was generated.* Global features record properties of the entire tree. Features and examples are discussed below.

Traversal features: For each node in the tree, features are generated that record the preorder traversal of the subtree rooted at that node, for all subtrees of all depths (up to the maximum depth). Feature names are constructed with the prefix "traversal-", followed by the concatenated names of the nodes (starting with the current node) on the traversal path. As an example, consider the sp-tree in Figure 6. Feature SP-TRAVERSAL-SOFT-MERGE*IMPLICIT-CONFIRM*IMPLICIT-CONFIRM has value 1, since it counts the number of subtrees in the sp-tree in which a soft-merge rule dominates two implicit-confirm nodes. In the DSyntS tree for alternative 8 (Figure 9), feature DSYNT-TRAVERSAL-PRONOUN, which counts the number of nodes in the DSyntS tree labelled PRONOUN (explicit or empty), has value 4.

Sister features: These features record all consecutive sister nodes. Names are constructed with the prefix "sisters-", followed by the concatenated names of the sister nodes. As an example, consider the sp-tree shown in Figure 8, and the DSyntS tree shown in Figure 9. Feature DSYNT-SISTERS-PRONOUN-ON1 counts the number of times the lexical items PRONOUN and ON1 are sisters in the DSyntS tree; its value is 1 in Figure 9. Another example is feature SP-SISTERS-IMPLICIT-CONFIRM*IMPLICIT-CONFIRM, which describes the configuration of all implicit confirms in the sp-trees in; its value is 2 for all three sp-trees in Figures 6, 7 and 8.

Ancestor features: For each node in the tree, these features record all the

*Lexicalized features could be useful in learning lexically specific restrictions on aggregation (for example, for verbs such as *meet*).

initial subpaths of the path from that node to the root. Feature names are constructed with the prefix "ancestor-", followed by the concatenated names of the nodes (starting with the current node). For example, the feature SP-ANCESTOR*IMPLICIT-CONFIRM-ORIG-CITY*SOFT-MERGE-GENERAL*SOFT-MERGE-GENERAL counts the number of times that two soft-merge-general nodes dominate an implicit confirm of the origin city; its value is 1 in the sp-trees of Figures 6 and 7, but 0 in the sp-tree of Figure 8.

Leaf features: These features record all initial substrings of the frontier of the sp-tree, which consists of elementary speech acts. Names are prefixed with "leaf-", and are then followed by the concatenated names of the frontier nodes (starting with the current node). The value is always 0 or 1. For example, the sp-trees of Figure 6, 7 and 8 have value 1 for features LEAF-IMPLICIT-CONFIRM AND LEAF-IMPLICIT-CONFIRM*IMPLICIT-CONFIRM, representing the first two sequences of speech acts on the leaves of the tree. Figure 6 sp-tree has value 1 for features LEAF-IMPLICIT-CONFIRM*IMPLICIT-CONFIRM*REQUEST, and LEAF-IMPLICIT-CONFIRM*IMPLICIT-CONFIRM*REQUEST*IMPLICIT-CONFIRM. Each of these has a corresponding parameterized feature, e.g. for LEAF-IMPLICIT-CONFIRM, there is a corresponding parameterized feature of LEAF-IMPLICIT-CONFIRM-ORIG-CITY.

Global Features: The global sp-tree features record, for each sp-tree and for each operation labeling a non-frontier node (i.e., rule such as CONJUNCTION or MERGE-GENERAL), (1) the minimal number of leaves (elementary speech acts) dominated by a node labeled with that rule in that tree (MIN); (2) the maximal number of leaves dominated by a node labeled with that rule (MAX); and (3) the average number of leaves dominated by a node labeled with that rule (AVG). For example, the sp-tree for alternative 8 in Figure 8 has value 2 for SOFT-MERGE-GENERAL-MAX -MIN, and -AVG, but a PERIOD-MAX of 5, PERIOD-MIN of 2 and PERIOD-AVG of 3.5.

5. Experimental Results

To train and test the **SPR** we partitioned the corpus into 5 disjoint folds and performed 5-fold cross-validation, in which at each fold, 80% of the examples were used for training an **SPR** and the other unseen 20% was used for testing. The folds were created by randomly choosing text plans that are input to SPOT. Thus each fold contains all the sentence plans for a given text plan. This method ensures that every example occurs once in the test set. We evaluate the performance of the trained **SPR** on the test sets of text plans by comparing for each text plan:

- BEST: The score of the top human-ranked sentence plan(s);
- SPOT: The score of SPOT's selected sentence plan;
- RANDOM: The score of a sentence plan randomly selected from the alternate sentence plans.

Figure 11 shows the cumulative distributions of scores for the highest ranked sp-tree for each of the 100 text plans, according to the human judges, according to SPoT, and according to random choice. Table 1 provides a summary of the means and standard deviations. The human rankings provide a topline for SPoT (since SPoT is choosing among options ranked by the humans, it cannot possibly do better), while the random scores provide a baseline. The BEST distribution shows that 97% of text plans had at least one sentence plan ranked 4 or better. The RANDOM distribution approximates the distribution of rankings for all sentence plans for all examples; the straight diagonal line shows that this is a normal distribution.

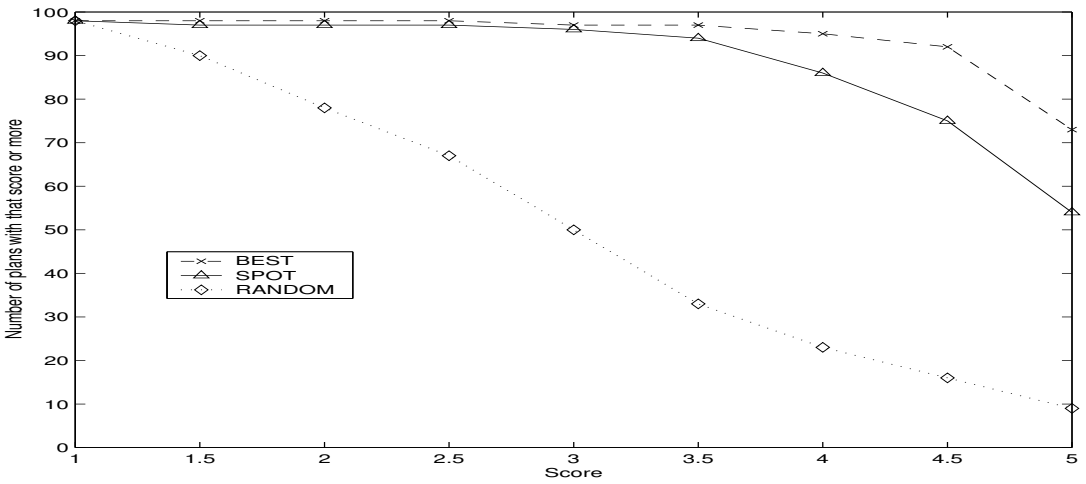

Figure 11: Distribution of rankings for BEST, SPOT and RANDOM

Features Used	Mean Score	S.D.
BEST	4.82	.40
SPoT	4.56	.68
RANDOM	2.76	.95

Table 1: Results, in comparison to topline and random baseline

Because each text plan is used in some fold of 5-fold cross validation as a test element, we assess the significance of the ranking differences with a paired t-test of SPOT to BEST and SPOT to RANDOM.

A paired t-test of SPOT to BEST shows that there are significant differences in performance ($t = 4.9, p < 0.005$). Perfect performance would have meant that there would be no significant difference. However, the mean of BEST is 4.82 as compared with the mean of SPOT of 4.56, for a mean difference of 0.26 on a

scale of 1 to 5 where scores closer to 5 are better. This is only a 5% difference in performance. Figure 11 also shows that the main differences are in the lower half of the distribution of rankings. The figure indicates that both BEST and SPoT have more than 50 out of 100 sentence plans with a score of 5. In other words, both distributions have a median of 5.

A paired t-test of SPOT to RANDOM shows that there are also significant differences in performance ($t = 18.2, p < 0.005$). The median of the RANDOM distribution is 2.50 as compared to SPoT's median of 5.0. The mean of RANDOM is 2.76, as compared to the mean of SPOT of 4.56, for a mean difference of 1.8 on a scale of 1 to 5. The performance difference in this case is 36%, showing a large difference in the performance of SPoT and RANDOM.

N	Condition	A0	A5	A8	α_s
1	LEAF-IMPLICIT-CONFIRM ≥ 1	1	1	1	0.94
2	DSYNT-TRAVERSAL-PRONOUN ≥ 2	1	2	4	-0.85
3	LEAF-IMPLICIT-CONFIRM*IMPLICIT-CONFIRM*REQUEST*IMPLICIT-CONFIRM ≥ 1	1	0	0	-0.52
4	DSYNT-TRAVERSAL-IN1 ≥ 1	0	0	1	-0.52
5	DSYNT-TRAVERSAL-PRONOUN ≥ 3	1	2	4	-0.34
6	SP-ANCESTOR*IMPLICIT-CONFIRM-ORIG-CITY*SOFT-MERGE-GENERAL*SOFT-MERGE-GENERAL ≥ 1.0	1	1	0	0.33
7	SP-ANCESTOR-SOFT-MERGE-GENERAL*PERIOD ≥ 1	0	1	0	0.21
8	DSYNT-ANCESTOR-IN1*LEAVE ≥ 1	0	0	1	-0.16
9	SP-TRAVERSAL-IMPLICIT-CONFIRM-DAY-NUMBER ≥ 1	1	1	1	-0.13
10	SP-TRAVERSAL-SOFT-MERGE*IMPLICIT-CONFIRM*IMPLICIT-CONFIRM ≥ 1	1	0	0	0.11
11	REL-CLAUSE-AVG ≥ 2	0	0	3	-0.12
12	PERIOD-AVG ≥ 3	0	5	3.5	0.12
13	DSYNT-ANCESTOR-TRAVEL*LIKE ≥ 1	1	0	0	0.10
14	DSYNT-SISTERS-PRONOUN-ON1 ≥ 1	0	1	1	-0.10
15	LEAF-IMPLICIT-CONFIRM*IMPLICIT-CONFIRM*REQUEST ≥ 1	1	0	0	-0.10
16	REL-CLAUSE-MIN ≥ 2	0	0	3	-0.09
17	SP-SISTERS-IMPLICIT-CONFIRM*IMPLICIT-CONFIRM- ≥ 1	2	2	2	0.09
18	REL-CLAUSE-MAX ≥ 2	0	0	3	-0.07
19	SP-ANCESTOR-IMPLICIT-CONFIRM*SOFT-MERGE*SOFT-MERGE-GENERAL ≥ 1	1	0	0	0.06

Figure 12: Rules with the largest impact on the final RankBoost score. α_s represents the increment or decrement associated with satisfying the condition. The columns A0, A5 and A8 give the values of the feature for alternatives 0, 5 and 8

We then examined the rules that SPoT learned in training and the resulting RankBoost scores. Figure 3 shows, for each alternative sentence plan, the BEST rating used as feedback to RankBoost and the score that RankBoost gave that example when it was in the test set in a fold. Recall that RankBoost focuses on learning *relative* scores, not absolute values, so the scores are normalized to range between 0 and 1.

Figure 12 shows some of the rules that were learned on the training data, that were then applied to the alternative sentence plans in each test set of each fold in order to rank them. We include only a subset of the rules that had the largest impact on the score of each sp-tree. We discuss some particular rule examples here to help the reader understand how SPoT's **SPR** works, but leave it to the reader to examine the thresholds and feature values in the remainder of the rules and sum the increments and decrements.

Rule (1) in Figure 12 states that an implicit confirmation as the first leaf of the sp-tree leads to a large (.94) increase in the score. Thus all three of our alternative sp-trees shown in Figures 6, 7 and 8 accrue this ranking increase. Rules (2) and (5) state that the occurrence of 2 or more PRONOUN nodes in the DSyntS reduces the ranking by 0.85, and that 3 or more PRONOUN nodes reduces the ranking by an additional 0.34. Alternative 8 is above the threshold for both of these rules; alternative 5 is above the threshold for Rule (2) and alternative 0 is always below the thresholds. Rule (6) on the other hand increases only the scores of alternatives 0 and 5 by 0.33 since alternative 8 is below the threshold for that feature.

Note also that the quality of the rules in general seems to be high. Although we provided multiple instantiations of features, some of which included parameters or lexical items that might identify particular discourse contexts, most of the learned rules utilize general properties of the sp-tree and the DSyntS. This is probably partly due to the fact that we eliminated features that appeared fewer than 10 times in the training data, but also partly due to the fact that boosting algorithms in general appear to be resistant to overfitting the data [12].

Features Used	Mean Score	S.D.
BEST	4.82	.40
ALL (=SPOT)	4.56	.68
DOMAIN-INDEPENDENT	4.55	.69
SP-DOMAIN-INDEPENDENT	4.52	.74
TASK-INDEPENDENT	4.20	.99
TASK-DEPENDENT	3.90	1.19
SP	4.41	.90
DSYNTS	4.13	1.17

Table 2: Results for subsets of features, and (for the sake of comparison), for BEST

In order to examine this issue further, we conducted a second set of experiments in which we partitioned the features and trained and tested SPoT on subsets of the original feature set. We were primarily interested in exploring the generalization potential of the **SPR** to other dialogue situations, but we were also interested in which types of features contributed most to performance. Thus we considered two orthogonal partitions of the features. The first partition divides the features into sentence-planning features (those from the sp-tree) and syntactic features (those from the DSyntS tree). The second partition divides the features into three sets according to their level of domain and task dependence. Domain independent features are features whose names include only closed-class lexical items, e.g. *in*, or names of the sentence plan tree operations, e.g. *merge*. Domain-dependent, task-independent features are those whose names include open class lexical items specific to this domain, e.g. *travel* or the names of the role slots, e.g. *destination city*. Domain dependent, task dependent features are features whose names include the value of a role filler for the domain, e.g. *Albuquerque*.

We ran a second set of experiments using these partitions, with results shown in Table 2. We compared the **SPR**'s performance at selecting a high ranking option for each feature partition shown in the table to the performance using all the features, and to other feature sets using t-tests with the modified Bonferroni statistic for multiple comparisons [47]. Because RankBoost uses a greedy algorithm, it is possible for a subset of a feature set to perform better than the whole feature set. The results indicated that the DOMAIN INDEPENDENT feature set (Row 3) performs as well as all the features ($t = .168$, $p = .87$), and that both the TASK INDEPENDENT ($t = 6.25$, $p < 0.00001$) and the TASK DEPENDENT ($t = 4.58$, $p < 0.00001$) feature sets perform worse. The sentence planning features SP also perform worse than all the features ($t = 2.54$, $p < .04$), but better than the DSyntS features ($t = p = 2.56$, $p < .04$). The DSyntS features perform worse than all the features ($t = 4.19$, $p < 0.00001$). The domain-independent subset of the sp features (SP-DOMAIN-INDEPENDENT) also performs as well as all the features ($t = .68$, $p = 1.0$).

6. Evaluation of SPoT

The evaluation discussed in the previous section shows that SPoT has indeed learned from the human judges. However, it does not show that using only two human judgments is sufficient to produce more broadly acceptable results, nor does it show that SPoT performs as well as optimized hand-crafted template or rule-based systems. In order to explore this issue, we conducted a second set of experiments to evaluate SPoT. Because SPoT is trained on data from AT&Ts DARPA Communicator system AMELIA, we can directly compare SPoT to the hand-crafted, template-based generation component of AMELIA. In order to perform an extensive 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 these different approaches in an experiment in which 60 human subjects rank the outputs of these different generators in the context of a spoken dialogue. An example output for each system for the text plan in Figure 2 is in Figure 13. We described SPoT above in detail and describe the RULE-BASED and BASELINE sentence planners in Sections 6.1 and 6.2.

The most important comparison is that between SPoT and the current generation component of AMELIA. Like most working research spoken dialogue systems, AMELIA 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 dialogue manager. The string templates for all the speech acts of a turn are heuristically ordered and then appended to produce the output. In order to produce output that is not highly redundant, string templates would need to be written for every possible combination of speech acts in a text plan. We refer to the output generated by AMELIA using this approach as the TEMPLATE output.

The experiment required human subjects to read five dialogues of real interactions with AMELIA. At 20 points over the five dialogues, AMELIA's actual utterance (TEMPLATE) is augmented with a set of variants; each set of variants included a representative generated by SPoT, and representatives of the four comparison sentence planners. 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 order of the sentences was randomized (though all subjects saw the same order). The subjects rated each variation on a 5-point Likert scale, by stating the degree to which they agreed with the statement *The system's utterance is easy to understand, well-formed, and appropriate to the dialogue context.* Sixty colleagues not involved in this research completed the experiment. In this evaluation technique the human subject is essentially an *overhearer* of the original conversation and makes judgements based on his or her overhearer status [7].

The remainder of this section describes the five sentence planners that we compare in more detail. SPoT, the two rule-based systems, and the two baseline sentence planners are all NLG based sentence planners. We described SPoT above. In all of the NLG sentence planners, each speech act is assigned a canonical lexico-structural representation (called a **DSyntS** – Deep Syntactic Structure [30]), as described for SPoT above. The basis of all the NLG systems are the clause-combining operations described in Section 3, and all of the NLG systems utilize the RealPro Surface realizer [21]. We exclude issues of lexical choice from this study, and restrict our attention to the question of how elementary structures for separate elementary speech acts are assembled into extended discourse.

System	Realization
TEMPLATE	Flying from Newark to Dallas, Leaving on the 1st of September, And what time did you want to leave?
SPoT	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	Leaving in September. Leaving on the 1st. What time would you, traveling from Newark to Dallas, like to leave?
NOAGG	Leaving on the 1st. Leaving in September. Going to Dallas. Leaving from Newark. What time would you like to leave?

Figure 13: Sample outputs for System5 of Dialogue D1 for each type of generation system used in the evaluation experiment.

Each of the sentence planners used in the evaluation experiment vary how the sp-tree is constructed. Section 6.1 describes the baselines, RANDOM and NOAGG. Section 6.2 describes the rule-based sentence planners, RBS and ICF.

6.1. Baseline Sentence Planners

In one obvious baseline system, the sp-tree is constructed by applying only the PERIOD operation: each elementary speech act is realized as its own sentence. This baseline, NOAGG, was suggested by Hovy and Wanner [16]. For NOAGG, we order the communicative acts from the text plan as follows: implicit confirms precede explicit confirms precede requests. Figure 13 includes a NOAGG output for the text plan in Figure 2.

The second baseline sentence planner simply applies combination rules randomly in the same way as the **SPG** described in Section 3, but stops when it has generated a single valid sentence plan. The resulting sentence planner we refer to as RANDOM. Figure 13 includes a RANDOM output for the text plan in Figure 2.

6.2. Two Rule-Based Sentence Planners

It has not been the object of our research to construct a rule-based sentence planner by hand, be it domain-independent or optimized for our domain. Our goal is to compare the SPOT 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 reasonably representative rule-based sentence planners. This

task was made easier by the fact that we could reuse much of the work done for SPoT, in particular the data structure of the sp-tree and the implementation of the clause-combining operations. We developed the two 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 dialogue systems. Since it was not clear which ordering would be optimal across all text plans, we constructed two rule-based systems that differ only in the initial ordering of the communicative acts in the input text plan.

In the first rule-based system, RBS (for "Rule-Based System"), we order the speech acts with explicit confirms first, then requests, then implicit confirms. Note that explicit confirms and requests do not co-occur in our data set. The second rule-based system is identical, except that implicit confirms come first rather than last. This system we call ICF (for "Rule-based System with Implicit Confirms First").

In the initial step of both RBS and ICF, we take the two leftmost members of the text plan and try to combine them using the following preference ranking of the combination operations: ADJECTIVE, the MERGES, CONJUNCTION, RELATIVE-CLAUSE, PERIOD. The first operation to succeed is chosen. This yields a binary sp-tree with three nodes, which becomes the **current sp-tree**. As long as the root node of the current sp-tree is not a PERIOD, we iterate through the list of remaining speech acts on the ordered text plan, combining each one with the current sp-tree using the preference-ranked operations as just described. The result of each iteration step is a binary, left-branching sp-tree. However, if the root node of the current sp-tree is a PERIOD, we start a new current sp-tree, as in the initial step described above. When the text plan has been exhausted, all partial sp-trees (all of which except for the last one are rooted in PERIOD) are combined in a left-branching tree using PERIOD. Cue words are added as follows: (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 [14]. Figure 13 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 with an example for which ICF works similarly. For RBS, the text plan in Figure 2 is ordered so that the request is first. For the request, the DSyntS 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-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 la-

beled 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, nor can we use MERGE or MERGE-GENERAL, since the verbs are not identical. Instead, we use SOFT-MERGE-GENERAL, which identifies 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?*.

6.3. Evaluation Results

All 60 subjects completed the experiment in a half hour or less. The experiment resulted in a total of 1200 judgments 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) SPOT, (3) two rule-based systems, and (4) two baseline systems. All statistically significant results discussed here had p values of less than .01.

System	Mean Score	S.D.
TEMPLATE	3.94	1.11
SPoT	3.88	1.27
BEST	3.76	1.29
RBS	3.38	1.43
ICF	3.50	1.43
No Aggregation	3.01	1.22
Random	2.66	1.45

Table 3: Summary of Overall Results for all Systems Evaluated

A comparison of the average ratings of BEST and SPOT in Table 1 with those in Table 3 shows that that the 60 human subjects gave overall lower ratings to BEST and SPOT than the two expert judges. This difference may be due to individual variation, or to the fact that the judges were explicitly attempting to use the full range of values when giving feedback for training. In any case, the fact that our learning method is based on ranking differences rather than absolute values means that only relative ranking is important.

We then turned to the question of whether differences in human ratings (score) were 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 Table 3.

As Table 3 indicates, some system outputs received more consistent scores than others, e.g. the standard deviation for TEMPLATE was much smaller than RANDOM. The ranking of the systems by average score is TEMPLATE, SPOT, ICF, RBS, NOAGG, and RANDOM. Posthoc comparisons of the scores of individual pairs of systems using the adjusted Bonferroni statistic revealed several different groupings.

The highest ranking systems were TEMPLATE and SPOT, whose ratings were not statistically significantly different from one another. This shows that it is possible to match the quality of a hand-crafted system with a trainable one, which should be 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 SPOT was statistically better than both of these systems ($p < .01$). Figure 14 shows that SPOT got more high rankings than either of the rule-based systems. In a sense this may not be that surprising, because as [16] point out, it is difficult to construct a rule-based sentence planner that handles all the rule interactions in a reasonable way. Features that SPoT's **SPR** uses allow SPOT 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, SPOT also scored better than the baseline systems NOAGG 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 SPOT to not be different than either system, e.g. if the sp-trees produced by RANDOM were all equally good, or if the aggregation rules that SPOT learned produced output less readable than NOAGG. Figure 14 shows that the distributions of scores for SPOT vs. the baseline systems are very different, with SPOT skewed towards higher scores.

Interestingly NOAGG also scored significantly better than RANDOM ($p < .01$), and the standard deviation of its scores was smaller (see Table 3). Remember that RANDOM's sp-trees often resulted in arbitrarily ordering the speech acts in the output. While NOAGG produced long redundant utterances, it placed the initiative taking speech act at the end of the utterance in its most natural

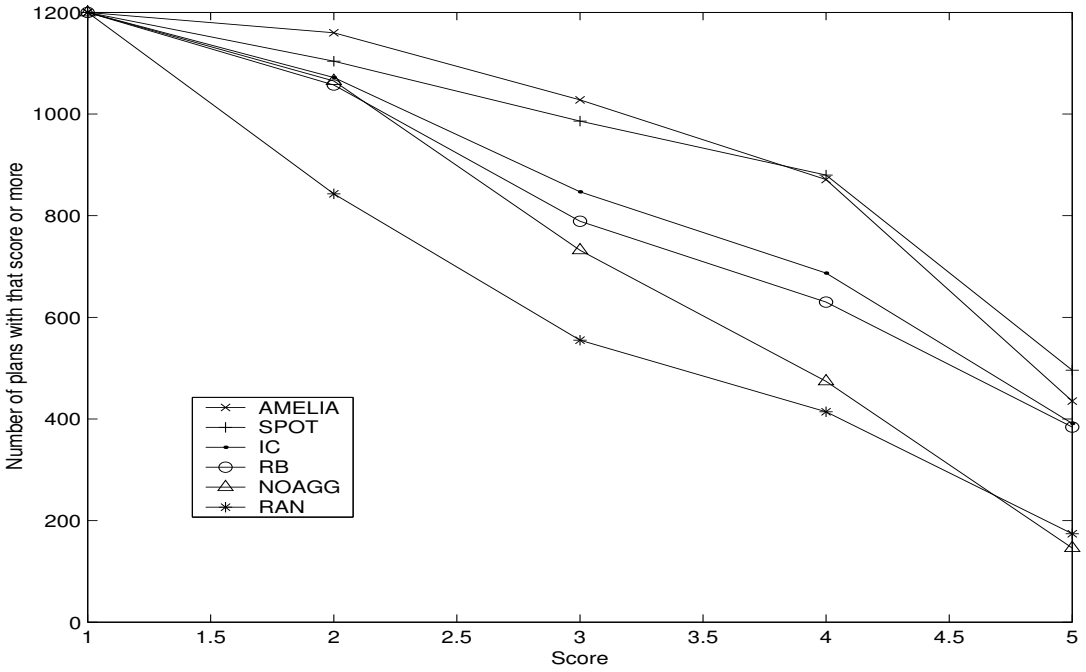

Figure 14: Chart comparing distribution of human ratings for SPoT, RBS, ICF, NoAgg and Random.

position, possibly resulting in a preference for NoAgg over Random. Another reason to prefer NoAgg could be its predictability.

7. Related Work

Machine learning approaches to natural language generation have only recently begun to be applied and there are many open issues with respect to the appropriate models and algorithms. To our knowledge, there is no other work reporting a machine learning approach to the problem of sentence scoping, but other work has explored automatically training other modules of a generator. A similar architure to the one we propose for sentence planning was suggested for stochastic generation in general by Oberlander and Brew [32]. There has been little work to date on using machine learning approaches for text planning. Duboue and McKeown describe algorithms to estimate content ordering constraints for descriptions in the medical domain [10]. Jordan and Walker applied rule induction to select the content of nominal expressions, comparing the output of the learner with what a human had originally said in a human-human dialogue [17].

Mellish *et. al* conducted a set of experiments using reinforcement learning (i.e. genetic algorithms) to determine the selection of utterances and their sequence when describing items in a museum collection [29]. This project combines aspects of text planning and sentence planning since content selection is part of text planning, and the linear ordering of a set of communicative goals is typically

considered a sentence planning problem. They showed that given an appropriate feedback function, this method could learn selection and sequencing rules, however they didn't evaluate the system's output by soliciting human judgments or comparing to human performance as we do here.

Other related work deals with discourse-related aspects of sentence planning such as cue word placement and selection [31, 11], clearly a crucial task whose integration into our approach we leave to future work.

There have also been a number of studies on using statistical methods for surface realization and prosody prediction. In surface realization, the focus has been on filtering a potential set of syntactic forms for a complete utterance using corpus probabilities to filter the possibilities [18, 20, 19, 1, 43], although there has also been research on selection of the form of a nominal expression using a classifier trained on a corpus of nominal expressions [35, 6]. Classifiers have also been trained on corpora labelled for TOBI accents to predict the appropriate prosody to output; these prosodic predictors have used various types of input features such as rhetorical structure, semantic features and syntactic features [15, 34].

In addition, some work on stochastic generation has been done within a template-based generation paradigm. Walker *et. al* use reinforcement learning to learn to select among a set of templates to achieve the communicative goals of summarizing or reading a set of email messages [46, 45]. Oh and Rudnicky [33] use n-gram models and Ratnaparkhi [37] uses maximum entropy to choose templates, using hand-written rules to score different candidates. Other work using reinforcement learning in spoken dialogue management focuses on selecting which of a set of communicative goals should be attempted at a particular state in the dialogue [25, 27, 41].

This paper also presented our evaluation of SPoT. Previous work on evaluation of natural language generation has utilized three different approaches to evaluation [28]. The first approach is a subjective evaluation methodology such as we use here, where human subjects rate NLG outputs produced by different sources [23, 2, 5]. Other work has evaluated template-based spoken dialogue generation with a task-based approach, i.e. the generator is evaluated with a metric such as task completion or user satisfaction after dialogue completion [45, 38]. 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 [48, 35]. When evaluating sentence planning, this approach doesn't make sense because many different realizations may be equally good. As mentioned above, this is the primary motivation for treating sentence planning

as a ranking problem, and it is possible that other generation problems would benefit by treatment as a ranking problem as well.

8. Discussion

We have presented SPoT, a trainable sentence planner. SPoT re-conceptualizes the sentence planning task as consisting of two distinct phases: (1) a very simple sentence plan generator **SPG** that generates multiple candidate sentence plans using weighted randomization; and (2) a sentence plan ranker **SPR** that can be trained from examples via human feedback, whose job is to rank the candidate sentence plans and select the highest ranked plan. Our results show that:

- SPoT's **SPR** selects sentence plans that on average are only 5% worse than the sentence plan(s) selected as the best by human judges.
- SPoT's **SPR** selects sentence plans that on average are 36% better than a random **SPR** that simply selects randomly among the candidate sentence plans.

We validated these results in an independent experiment in which 60 subjects evaluated the quality of different realizations for a given turn. (Recall that our trainable sentence planner was trained on the scores of only two human judges.) To our knowledge, this is the first reported experimental comparison of a trainable technique that shows that the quality of system utterances produced with trainable components can compete with hand-crafted or rule-based techniques. This evaluation revealed that the choices made by SPoT were not statistically distinguishable from the choices ranked at the top by the two human judges. More importantly, they were also not statistically different from the current hand-crafted template-based output of the AT&T Communicator system, which has been developed and fine-tuned over an extended period of time (whereas SPoT is based on judgments that took about three person-days to make). In addition, we expect SPoT to be more easily and quickly tuned to a new domain than template-based generation: the training materials for the SPoT sentence planner can be collected from subjective judgments from a small number of judges with little or no linguistic knowledge. The evaluation also showed that SPoT was rated better than two rule-based versions of our **SPG** which we developed as baselines. All systems outperformed the random choice.

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 template and rule-based systems, any quality standard 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 dialogue 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. We hope to extend SPoT and integrate it into AMELIA.

In future work, we intend to build on the work reported in this paper in several ways. First, we believe that we could utilize additional features as predictors of the quality of a sentence plan. These include features based on the discourse context, and features that encode relationships between the sp-tree and the DSyntS. We will also expand the capabilities of the **SPG** to cover additional sentence planning tasks in addition to sentence scoping, and duplicate the methods described here to retrain SPoT for our extended **SPG**.

9. Acknowledments

We would like to thank Michael Collins and Rob Schapire for their help, comments, and encouragement, and Noemie Elhadad and three anonymous reviewers for very useful feedback. This work was partially funded by DARPA under contract MDA972-99-3-0003. This work was completed while the second author was at AT&T Labs Research.

References

[1] Srinivas Bangalore and Owen Rambow. Exploiting a probabilistic hierarchical model for Generation. In *COLING*, Saarbucken, Germany, 2000.

[2] Srinivas Bangalore, Owen Rambow, and Steve Whittaker. Evaluation metrics for generation. In *Proceedings of the First International Natural Language Generation Conference (INLG2000)*, Mitzpe Ramon, Israel, 2000.

[3] M. Beutnagel, A. Conkie, J. Schroeter, Y. Stylianou, and A. Syrdal. The AT&T next-generation text-to-speech system. In *Meeting of ASA/EAA/DAGA in Berlin, Germany*, 1999.

[4] I. Bulyko and M. Ostendorf. Joint Prosody Prediction and Unit Selection for Concatenative Speech Synthesis. In *ICASSP 2001*, 2001.

[5] Charles Callaway and James Lester. Narrative prose generation. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, 2001.

[6] Hua Cheng, Massimo Poesio, Renate Henschel, and Chris Mellish. Corpus-based np modifier generation. In *Proceedings of the North American Meeting of the Association for Computational Linguistics*, 2001.

[7] Herbert H. Clark and Deanna Wilkes-Gibbs. Referring as a collaborative process. *Cognition*, 22:1–39, 1986.

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

[9] Laurence Danlos. 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, 2000.

[10] P. A. Duboue and McKeown K. R. Empirically estimating order constraints for content planning in generation. In *Proceedings of the 39rd Annual Meeting of the Association for Computational Linguistics (ACL/EACL-2001)*, 2001.

[11] Barbara Di Eugenio, Johanna D. Moore, and Massimo Paolucci. Learning features that predict cue usage. In *Proceedings of the 35th Annual Meeting of the Association of Computational Linguistics, ACL/EACL 97*, 1997.

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

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

[14] Julia Hirschberg and Diane Litman. Empirical studies on the disambiguation of cue phrases. *Computational Linguistics*, 19(3):501–530, 1993.

[15] Janet Hitzeman, Alan W. Black, Paul Taylor, Chris Mellish, and Jon Oberlander. On the use of automatically generated discourse-level information in a concept-to-speech synthesis system. In *Proceedings of the International Conference on Spoken Language Processing, ICSLP98*, 1998.

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

[17] Pamela Jordan and Marilyn A. Walker. Learning attribute selections for non-pronominal expressions. In *In Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-00), Hong Kong*, 2000.

[18] Kevin Knight and V. Hatzivassiloglou. Two-level, many-paths generation. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL'95)*, pages 252–260, 1995.

[19] I. Langkilde. Forest-based statistical sentence generation. In *In Proceedings of the 6th Applied Natural Language Processing Conference and the 1st Meeting of the North American Chapter of the Association of Computational Linguistics (ANLP-NAACL 2000)*, pages 170–177, 1998.

[20] I. Langkilde and K. Knight. Generation that exploits corpus-based statistical knowledge. In *Proceedings of COLING-ACL*, 1998.

[21] Benoit Lavoie and Owen Rambow. A fast and portable realizer for text generation systems. In *Proceedings of the Third Conference on Applied Natural Language Processing, ANLP97*, pages 265–268, 1997.

[22] Benoit Lavoie and Owen Rambow. A framework for customizable generation of multi-modal presentations. In *COLING-ACL98*, Montréal, Canada, 1998. ACL.

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

[24] W. J. M. Levelt. *Speaking: From Intention to Articulation.* MIT Press, 1989.

[25] E. Levin and R. Pieraccini. A stochastic model of computer-human interaction for learning dialogue strategies. In *EUROSPEECH 97*, 1997.

[26] Esther Levin, S Narayanan, R. Pieraccini, K. Biatov, E. Bocchieri, G. Di-Fabbrizio, W. Eckert, S. Lee, A. Pokrovsky, M. Rahim, P. Ruscitti, and M. Walker. The AT&T DARPA communicator mixed-initiative spoken dialog system. In *Proceedings of the International Conference on Spoken Language Processing,ICSLP00*, 2000.

[27] Diane J. Litman, Michael S. Kearns, Satinder Singh, and Marilyn A. Walker. Automatic optimization of dialogue management. In *Proc. COLING 2000*, 2000.

[28] Chris Mellish and Robert Dale. Evaluation in the context of natural language generation. *Computer Speech and Language*, 12(3), 1998.

[29] Chris Mellish, Alistair Knott, Jon Oberlander, and Mick O'Donnell. Experiments using stochastic search for text planning. In *Proceedings of International Conference on Natural Language Generation*, pages 97–108, 1998.

[30] Igor A. Melčuk. *Dependency Syntax: Theory and Practice.* SUNY, Albany, New York, 1988.

[31] Margaret G. Moser and Johanna Moore. Investigating cue selection and placement in tutorial discourse. In *ACL 95*, pages 130–137, 1995.

[32] Jon Oberlander and Chris Brew. Stochastic text generation. *Philosophical Transactions of the Royal Society of London*, Series A, 358:1373–1385, 2000.

[33] Alice H. Oh and Alexander I. Rudnicky. Stochastic language generation for spoken dialog systems. In *Proceedings of the ANL/NAACL 2000 Workshop on Conversational Systems*, pages 27–32, Seattle, 2000. ACL.

[34] Shimei Pan and Kathleen McKeown. Learning Intonation Rules for Concept to Speech Generation. In *COLING-ACL98*, pages 1003–1009, Montreal, Canada, 1998.

[35] Massimo Poesio. Annotating a corpus to develop and evaluate discourse entity realization algorithms: issues and preliminary results. In *Proc. Language Resources and Evaluation Conference, LREC-2000*, 2000.

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

[37] Adwait Ratnaparkhi. Trainable methods for surface natural language generation. In *Proceedings of First North American ACL*, Seattle, USA, May 2000.

[38] Ehud Reiter, R. Robertson, S Lennox, , and L Osman. Using a randomised controlled clinical trial to evaluate an nlg system. In *Proceedings of ACL-2001*, pages 434–441, 2001.

[39] Robert E. Schapire. A brief introduction to boosting. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 1999.

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

[41] Satinder Singh, Michael S. Kearns, Diane J. Litman, and Marilyn A. Walker. Empirical evaluation of a reinforcement learning spoken dialogue system. In *Proc. AAAI 2000*, 2000.

[42] Matthew Stone and Christine Doran. Sentence planning as description using tree adjoining grammar. In *Proceedings of the 35th Annual Meeting of the Association of Computational Linguistics, ACL/EACL 97*, pages 198–205, Madrid, Spain, 1997.

[43] Sebastian Varges. Instance-based natural language generation. In *Proceedings of the North American Meeting of the Association for Computational Linguistics*, 2001.

[44] M. Walker, J. Aberdeen, J. Boland, E. Bratt, J. Garofolo, L. Hirschman, A. Le, S. Lee, S. Narayanan, K. Papineni, B. Pellom, J. Polifroni, A. Potamianos, P. Prabhu, A. Rudnicky, G. Sanders, S. Seneff, D. Stallard, and S. Whittaker. DARPA Communicator dialog travel planning systems: The june 2000 data collection. In *EUROSPEECH 2001*, 2001.

[45] Marilyn A. Walker. An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email. *Journal of Artificial Intelligence Research*, 12:387–416, 2000.

[46] Marilyn A. Walker, Jeanne C. Fromer, and Shrikanth Narayanan. Learning optimal dialogue strategies: A case study of a spoken dialogue agent for email. In *Proceedings of the 36th Annual Meeting of the Association of Computational Linguistics, COLING/ACL 98*, pages 1345–1352, 1998.

[47] S. Wilks. *Mathematical Statistics*. Wiley, 1962.

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