SPoT: A Trainable Sentence Planner

Marilyn A. Walker
AT&T Labs – Research
Florham Park, NJ, USA
walker@research.att.com

Owen Rambow
AT&T Labs – Research
Florham Park, NJ, USA
rambow@research.att.com

Monica Rogati
Carnegie Mellon University
Pittsburgh, PA, USA
mrogati+@cs.cmu.edu

Abstract

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. In this paper, we present **SPoT**, a sentence planner, and a new methodology for automatically training **SPoT** on the basis of feedback provided by human judges. We reconceptualize the task into two distinct phases. First, a very simple, randomized sentence-plan-generator (**SPG**) *generates* a potentially large list of possible sentence plans for a given text-plan input. Second, the sentence-plan-ranker (**SPR**) *ranks* the list of output sentence plans, and then selects the top-ranked plan. The **SPR** uses ranking rules automatically learned from training data. We show that the trained **SPR** learns to select a sentence plan whose rating on average is only 5% worse than the top human-ranked sentence plan.

1 Introduction

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 sentences.[1] For example, consider the required capabilities of a sentence planner for a mixed-initiative spoken dialog system for travel planning:

(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, 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. In User4, the caller provides this information but also provides 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 1. 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 2.[2]

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

Figure 1: The text plan (communicative goals) for utterance System5 in dialog 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
8	Leaving in September. Leaving on the 1st. What time would you, traveling from Newark to Dallas, like to leave?	2	.39

Figure 2: Alternative sentence plan realizations for the text plan for utterance System5 in dialog D1. H = human rating, RB = RankBoost score.

In this paper, we present **SPoT**, for "Sentence Planner, Trainable". We also present a new methodology for *automatically training* **SPoT** on the basis of feedback provided by human judges. In order to train **SPoT**, we reconceptualize its task as consisting of two distinct phases. In the first phase, the sentence-plan-generator

[1] 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.

[2] The meaning of the human ratings and RankBoost scores in Figure 2 are discussed below.

(**SPG**) *generates* a potentially large sample of possible sentence plans for a given text-plan input. 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. Our primary contribution is a method for training the **SPR**. The **SPR** uses rules automatically learned from training data, using techniques similar to (Collins, 2000; Freund et al., 1998).

Our method 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. We show that the trained **SPR** learns to select a sentence plan whose rating on average is only 5% worse than the top human-ranked sentence plan. 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 in section 5.

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. In our case, the output of the dialog manager of a spoken dialog system provides the input to our sentence planner in the form of a single spoken dialog text plan for each of the turns. (In contrast, the dialog managers of most dialog systems today simply output completely formed utterances which are passed on to the TTS module.) Each text plan is an unordered set of elementary speech acts encoding all of the system's communicative goals for the current turn, as illustrated in Figure 1. Each elementary speech act is represented as a type (request, implicit confirm, explicit confirm), with type-specific parameters. The sentence planner must decide among alternative abstract linguistic resources for this text plan; surface realizations of some such alternatives are in Figure 2.

As already mentioned, we divide the sentence planning task into two phases. In the first phase, the sentence-plan-generator (**SPG**) generates 12-20 possible sentence plans for a given input text plan. Each speech act is assigned a canonical lexico-structural representation (called a **DSyntS** – Deep Syntactic Structure (Mel'čuk, 1988)). 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 sentence plans generated by the **SPG**, and then selects the top-ranked output as input to the surface realizer, RealPro (Lavoie and Rambow, 1997). The architecture is summarized in Figure 3.

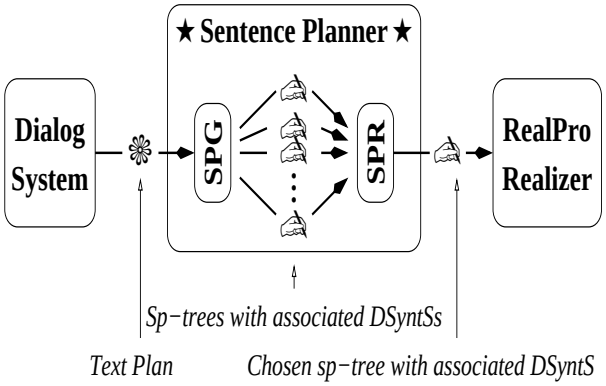

Figure 3: Architecture of **SPoT**

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 lexico-structural representation, by combining these representations using the following combining operations. Examples can be found 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.

These operations are not domain-specific and are similar to those of previous aggregation components (Rambow and Korelsky, 1992; Shaw, 1998; Danlos, 2000), although the various MERGE operations are, to our knowledge, novel in this form.

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

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

from the input text plan, and with its interior nodes labeled with clause-combining operations[3]. 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 by executing their clause-combing 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.

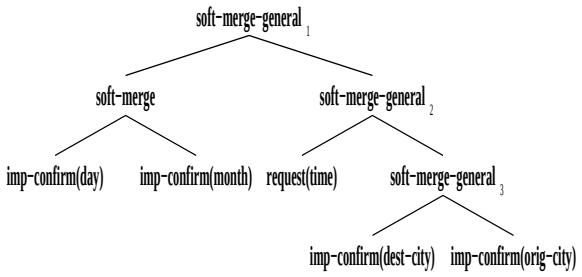

Figure 5: Alternative 0 Sentence Plan Tree

Figure 2 shows some of the realizations of alternative sentence plans generated by our **SPG** for utterance System5 in Dialog D1. Sp-trees for alternatives 0, 5 and 8 are in Figures 5, 6 and 7. For example, consider the sp-tree in Figure 7. Node **soft-merge-general** merges an implicit-confirmations 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 8 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 7), and the diagram also shows how structures are composed into larger structures by the clause combining operations.

[3]The sp-tree is inspired by (Lavoie and Rambow, 1998). The representations used by Danlos (2000), Gardent and Webber (1998), or Stone and Doran (1997) are similar, but do not (always) explicitly represent the clause combining operations as labeled nodes.

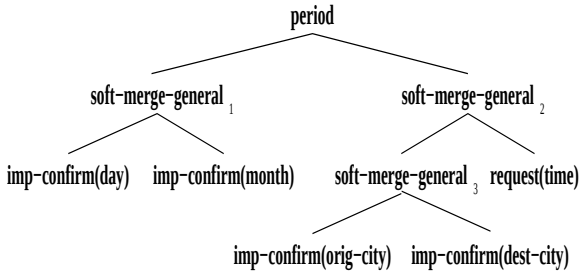

Figure 6: Alternative 5 Sentence Plan Tree

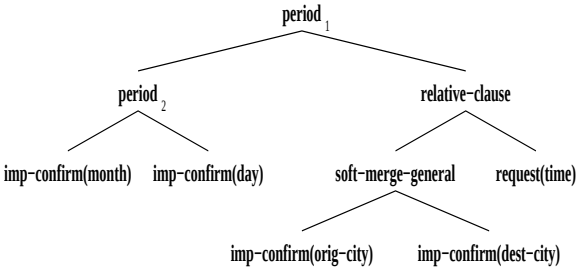

Figure 7: Alternative 8 Sentence Plan Tree

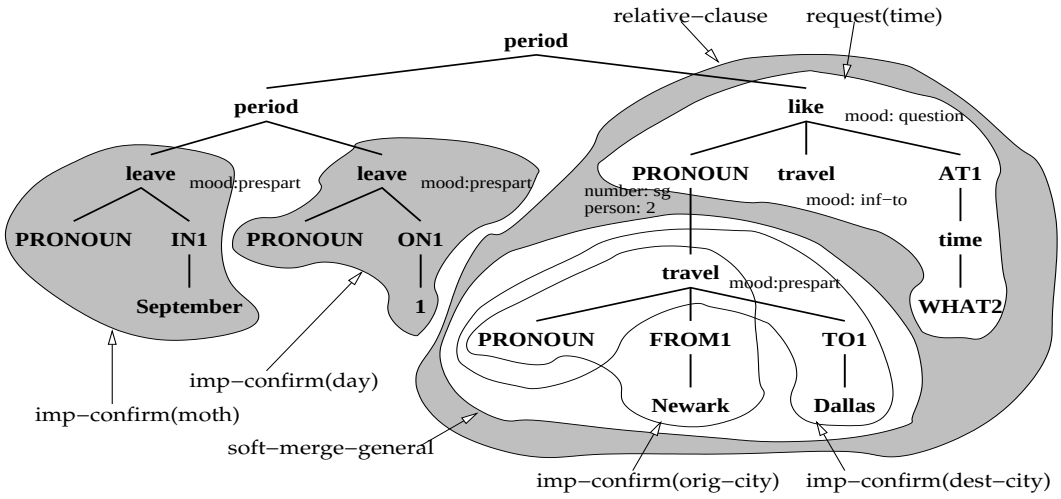

Figure 8: Alternative 8 DSyntS (not all linguistic features are shown)

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 sentence plans, by randomly selecting among the operations according to some probability distribution.[4]

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 (Freund et al., 1998), to learn from a labelled set of sentence-plan training examples a set of rules for scoring sentence plans.

4.1 RankBoost

RankBoost is a member of a family of boosting algorithms (Schapire, 1999). Freund et al. (1998) describe the boosting algorithms for ranking in detail: for completeness, we give a brief description in this section. 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 4.2. 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 plausibility. The training examples are used to set the parameter values α_s. In (Freund et al., 1998) 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.

Freund et al. (1998) then describe training as a 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 most impact on the loss function – and updates the parameter value to minimize the loss.

[4] 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.

The result is that substantial progress is typically made in minimizing the error rate, with relatively few non-zero parameter values. Freund et al. (1998) 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. Empirical results for boosting have shown that in practice the method is highly effective.

4.2 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 dialogs by the dialog system. 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 expert judges in the context of the transcribed original dialogs (and therefore also with respect to their adequacy given the communicative goals for that turn), 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. The ratings assigned to the sentence plans were roughly normally distributed, with a mean of 2.86 and a median of 3. Each sp-tree provided an example input to RankBoost, and each corresponding rating was the feedback for that example.

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 Collins (2000). The motivation for the features was to capture declaratively decisions made by the randomized **SPG**. 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 dialog in which the text plan was generated.[5] 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 5. 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 8), 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 7, and the DSyntS tree shown in Figure 8. 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 8. 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 5, 6 and 7.

Ancestor features: For each node in the tree, these features record all the 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 5 and 6, but 0 in the sp-tree of Figure 7.

Leaf features: These features record all initial substrings of the frontier of the sp-tree (recall that its frontier 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 5, 6 and 7 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 5 sp-tree has value 1 for features LEAF-IMPLICIT-CONFIRM*IMPLICIT-CONFIRM*REQUEST, and LEAF-IMPLICIT-

[5] Lexicalized features are useful in learning lexically specific restrictions on aggregation (for example, for verbs such as *kiss*).

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 9: 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

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 7 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. This method ensures that every example occurs once in the test set. We evaluate the performance of the 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 10 shows the distributions of scores for the highest ranked sp-tree for each of the 100 text plans, according to the human experts, according to **SPoT**, and according to random choice. 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.

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. This is only a 5% difference in performance. Figure 5 also shows that the main differences are in the lower half of the distribution of rankings; 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 distri-

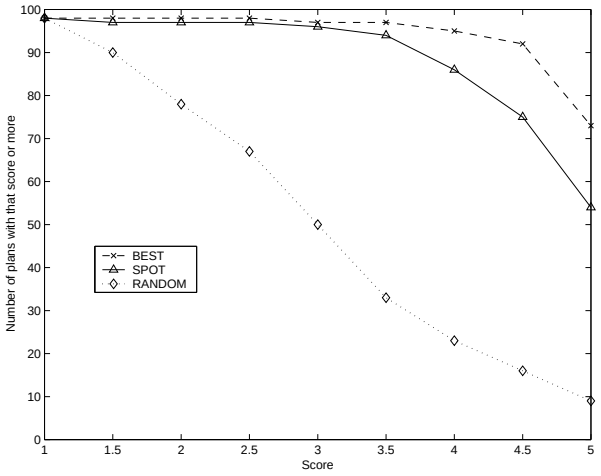

Figure 10: Distribution of rankings for BEST, SPOT and RANDOM

bution 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.

We then examined the rules that **SPoT** learned in training and the resulting RankBoost scores. Figure 2 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 9 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 9 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 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 (Freund et al., 1998).

6 Related Work

Previous work in sentence planning in the natural language generation (NLG) community uses hand-written rules to approximate the distribution of linguistic phenomena in a corpus (see (Shaw, 1998) for a recent example with further references). This approach is difficult to scale due to the nonrobustness of rules and unexpected interactions (Hovy and Wanner, 1996), and it is difficult to develop new applications quickly. Presumably, this is the reason why dialog systems to date have not used this kind of sentence planning.

Most dialog systems today use template-based generation. The template outputs are typically concatenated to produce a turn realizing all the communicative goals. It is hard to achieve high quality output by concatenating the template-based output for individual communicative goals, and templates are difficult to develop and maintain for a mixed-initiative dialog system. For these reasons, Oh and Rudnicky (2000) use n-gram models and Ratnaparkhi (2000), maximum entropy to choose templates, using hand-written rules to score different candidates. But syntactically simplistic approaches may have quality problems, and more importantly, these approaches only deal with inform speech acts. And crucially, these approaches suffer from the need for training data. In general there may be no corpus available for a new application area, or if there is a corpus available, it is a transcript of human-human dialogs. Human-human dialogs, however, may not provide a very good model of sentence planning strategies for a computational system because the sentence planner must plan communicative goals such as *implicit confirmation* which are needed to prevent and correct errors in automatic speech recognition but which are rare in human-human dialog.

Other related work deals with discourse-related aspects of sentence planning such as cue word placement (Moser and Moore, 1995), clearly a crucial task whose integration into our approach we leave to future work. Mellish et al. (1998) investigate the problem of determining a discourse tree for a set of elementary speech acts which are partially constrained by rhetorical relations. Using hand-crafted evaluation metrics, they show that a genetic algorithm achieves good results in finding discourse trees. However, they do not address clause-combining, and we do not use hand-crafted metrics.

7 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.) 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 distinguishable statistically 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). **SPoT** also was rated better than two rule-based versions of our **SPG** which we developed as baselines. All systems outperformed the random choice. We will report on these results in more detail in a future publication.

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**.

References

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.

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

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. 1997. RealPro – a fast, portable sentence realizer. In *Proceedings of the Conference on Applied Natural Language Processing (ANLP'97)*, Washington, DC.

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.

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

Chris Mellish, Alistair Knott, Mick O'Donnell, and Jon Oberlander. 1998. Experiments using stochastic search for text planning. In *Proceedings of the 8th International Workshop on Natural Language Generation*, pages 98–107, Niagara-on-the-Lake, Ontario.

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

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

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

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

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

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.