

A Trainable Generator for Recommendations in Multimodal Dialog

Marilyn Walker, Rashmi Prasad and Amanda Stent

University of Sheffield, University of Pennsylvania, SUNY Stony Brook
walker@dcs.shef.ac.uk, rjprasad@linc.cis.upenn.edu, stent@cs.sunysb.edu

Abstract

As the complexity of spoken dialogue systems has increased, there has been increasing interest in spoken language generation (SLG). SLG promises portability across application domains and dialogue situations through the development of application-independent linguistic modules. However in practice, rule-based SLGs often have to be tuned to the application. Recently, a number of research groups have been developing hybrid methods for spoken language generation, combining general linguistic modules with methods for training parameters for particular applications. This paper describes the use of boosting to train a sentence planner to generate recommendations for restaurants in MATCH, a multimodal dialogue system providing entertainment information for New York.

1. Introduction

In the past few years, as the complexity of spoken dialogue systems has increased, there has been increasing interest in spoken language generation (SLG) [10]. In contrast to pre-recorded speech or template-based generation for system output, SLG takes as input a conceptual representation of the content that the system wants to communicate and outputs a (possibly marked up) string to the Text-To-Speech (TTS) component [13]. SLG promises improved system output, portability and customizability to different application domains, dialogue situations, and users, through a combination of application-independent linguistic modules [5], and methods for automatically training parameters for particular applications [7, 12, 4, 11].

strategy:	recommend
items:	Bar Pitti, Arlecchino, Babbo, Cent'anni, Cucina Stagionale, Grand Ticino, Il Mulino, John's Pizzeria, Marinella, Minetta Tavern, Trattoria Spaghetti, Vittorio Cucina
relations:	justify(nuc:1;sat:2); justify(nuc:1;sat:3); justify(nuc:1;sat:4)
content:	1. assert(best(Babbo)) 2. assert(has-att(Babbo, foodquality(superb))) 3. assert(has-att(Babbo, decor(excellent))) 4. assert(has-att(Babbo, service(excellent)))

Figure 1: Text Plan for a recommendation for an Italian restaurant in the West Village

This paper describes a trainable sentence planner for information presentation in the MATCH multimodal dialogue system [3]. Information presentation in MATCH focusses on the generation of user-tailored *recommendations*, *comparisons* and *summaries* [9]. The goal of these dialogue strategies is to improve dialogue efficiency and user satisfaction by only providing information predicted to be most relevant to the user.

We focus here on the generation of recommendations. The input to the sentence planner for each recommendation is a *text plan*, which is a set of *assertions* of facts about a set of restaurants that the user is considering, as well as a specification of the *rhetorical relations* that hold between those facts that must be communicated as well. Each rhetorical relation designates one or more facts as the *nuclei* of the relation, i.e. the main point, and the other facts as *satellites*, i.e. the supplementary

Alt	Realization	Human	RB
2	Babbo has excellent service. It has superb food quality. It has excellent decor. It has the best overall quality among the selected restaurants.	2	0.21
10	Since Babbo has excellent service and superb food quality, with excellent decor, it has the best overall quality among the selected restaurants.	3.5	0.77
12	Babbo has the best overall quality among the selected restaurants because it has superb food quality, with excellent service, and it has excellent decor.	1.5	0.45
16	With excellent decor, excellent service and superb food quality, Babbo has the best overall quality among the selected restaurants.	5	0.91
20	Babbo has excellent service and superb food quality, with excellent decor. It has the best overall quality among the selected restaurants.	4	0.88

Figure 2: Some of the Alternative Sentence Plan Realizations for the Recommend Plan in Figure 1.

facts [6]. For example, in the recommendation text plan in Figure 1, the rhetorical relation *justify(nuc:1;sat:2)* expresses that the assertion *Babbo has superb food quality* provides (factual) information supporting the assertion *Babbo has the best overall quality*. The job of the sentence planner is to choose linguistic resources to realize a text plan and then rank the resulting alternative realizations.

Previous work describes SPoT (Sentence Planner Trainable), a sentence planning module for the AT&T Communicator system that was automatically trained on the basis of human feedback [11]. However, the training methodology embodied in SPoT was only shown to work for the information gathering phase of the dialogue, where text plans were simpler because rhetorical relations did not hold among the concepts to be communicated. This paper describes an extension of SPoT, dubbed SPaRky (Sentence Planning with Rhetorical Knowledge), that operates on more complex text plans. Figure 2 illustrates some of the many potential realizations that SPaRky produces for the recommend plan in Figure 1 (the Human and RB columns are explained below).

Section 2 describes the overall architecture of SPaRky. Like SPoT, SPaRky is a trainable sentence planner, consisting of two modules: (1) a randomized sentence plan generator (SPG) (Section 3); and (2) a sentence plan ranker (SPR) that is trained to rank the candidate sentence plans produced by the SPG on the basis of human feedback (Section 4). Section 5 presents the results of a training experiment. We show that given input such as Figure 1, SPaRky outputs sentence plans that: (1) communicate the desired rhetorical relations; (2) are significantly better than a randomly selected sentence plan; and (3) are on average only 6% worse than the sentence plan ranked highest by human labellers.

2. Generator Architecture

The architecture of the spoken language generator for MATCH is a series of pipelined modules. See Figure 3. The dialogue manager passes a high-level communicative goal to the SPUR text planner (See [9]), which selects the content to be communicated. This content is specified in a text-plan consisting of a set

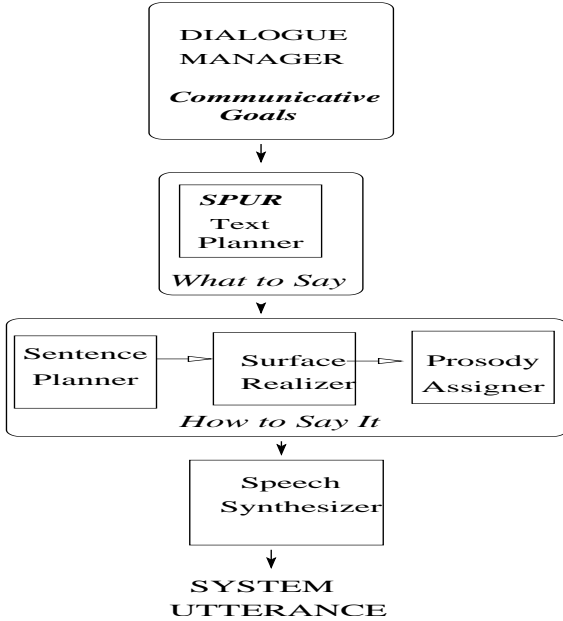


Figure 3: Generator Architecture in MATCH.

of speech-acts to be communicated and the rhetorical relations that hold between them as in Figure 1 [7].

The sentence planner consists of three separate modules. First, a content structuring module operates on the input text plan to determine one or more ways to linearly order and hierarchically group the elementary speech acts. In order to generate a population of sentence plans that are reasonably good, the algorithm for content structuring utilizes (a) principles of rhetorical structure [6] that either create complex rhetorical structures from simpler ones or transform one rhetorical structure into another [2], and (b) principles of entity-based coherence [1]. For example, the algorithm can combine multiple mononuclear relations having the same nucleus into one complex mononuclear relation with a combined satellite, and it prefers linear orderings that maximize the *principle of continuity* [4]. The result of the content structuring module is encoded in a text-plan tree (**tp-tree**), of which there may be one or more, with leaves labelled by speech acts and interior nodes labelled by rhetorical relations. For example, one of the generated tp-trees for the plan in Figure 1 is in Figure 4. The *infer* relation is used to join multiple satellites in a mononuclear relation or the nuclei in a multinuclear relation.¹

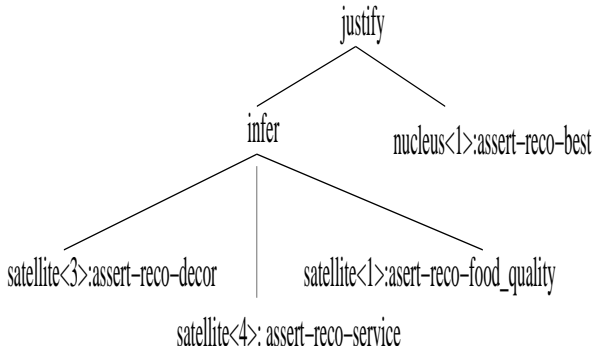


Figure 4: A text plan tree for the recommend plan in Figure 1.

¹While the *infer* relation is similar to the *joint* relation in RST, the RST *joint* holds only in a multinuclear relation.

Then, the sentence plan generator (SPG) operates on the set of tp-trees, choosing the linguistic means for achieving the communicative goals. This results in a set of sentence plan trees (**sp-trees**), which are passed to the sentence plan ranker. The sentence plan ranker (SPR) uses a set of rules for ranking sentence plans that were learned from a labelled set of sentence-plan training examples using the RankBoost algorithm, a type of boosting algorithm [8]. The column labelled Human in Figure 2 shows the Human feedback used in training the SPR, and the RB column shows the ranking for these examples that the SPR learned in the experiments described in Section 5. After ranking the generated sp-trees, the SPR then selects the top-ranked plan as input to the surface realizer, RealPro [5]. RealPro transforms the linguistic structures into a surface linguistic utterance by adding function words, inflecting words and determining word order. The prosody assignment module then uses the prior levels of linguistic representation to determine the appropriate prosody, before passing a marked-up string to the TTS module. (See Figure 3.)

3. Sentence Plan Generation

The basis of the SPG is a set of clause-combining operations that operate on the elementary predicate-argument representations (lexico-structural representations called **DSyntS** – [5]) associated with each primitive speech act on the leaves of the tp-tree. The application of the operations is bottom-up and left-to-right, and is restricted by the rhetorical relation holding at the interior nodes. . . It results in two parallel structures: (1) the **sp-tree**, a binary tree with leaves labeled by the speech acts from the input text plan, and interior nodes labeled with clause-combining operations; and (2) one or more DSyntS trees (**d-trees**) which reflect the parallel operations on the predicate-argument representations. We generate a random sample of possible sentence plans for each text plan, up to a pre-specified sample size, by randomly selecting among the operations according to a probability distribution that favors preferred operations. The clause-combining operations are:²

- **MERGE**: Two clauses can be combined if they have identical matrix verbs and identical arguments and adjuncts except one. The non-identical arguments are coordinated. Used with the relation *INFER*.
- **WITH-REDUCTION**: Two clauses with identical subject arguments can be identified if one of the clauses has a *HAVE*-possession matrix verb. The possession clause undergoes *with*-participial clause formation and is attached to the non-reduced clause. Used with the relations *JUSTIFY* and *INFER*.
- **RELATIVE-CLAUSE**: Two clauses with an identical subject can be identified. One clause is attached to the subject of the other clause as a relative clause. Used with the relations *JUSTIFY* and *INFER*.
- **CUE-WORD-CONJUNCTION**: Two clauses are conjoined with a cue word (coordinating or subordinating conjunction). Choice of cue word is sensitive to the rhetorical relation.
- **CUE-WORD-INSERTION**: Two clauses are joined with a period and a cue word is inserted in the second clause. Choice of cue word is sensitive to the rhetorical relation.
- **PERIOD**: Two clauses can be joined by a period.

Figure 5 provides examples of these operations. For each operation, the figure specifies the rhetorical relation the operation is being used for in this example. Each combining operation can switch the order of its arguments, from satellite before nucleus to nucleus before satellite. Figure 2 shows some of the utterances generated by the SPG for the text plan in Figure 1. Note that the same tp-tree can have radically different realizations, as determined by the operations of the SPG, e.g. after the sentence planning phase, the tp-tree in Figure 4 yields both Alt 2 and Alt 16 in Figure 2. The sp-tree for Alt 16 is in Figure 6. The sp-tree indicates that the *MERGE* operation was used to realize the *INFER* rhetorical relation, and that the *WITH-REDUCTION* operation was applied with the order of satellite before nucleus

²Cue words used by the SPG for recommendations were *because*, *since*, and *and*.

Rule	Relation	Sample first argument	Sample second argument	Result
MERGE	INFER	Babbo has superb decor.	Babbo has mediocre food quality.	Babbo has superb decor and mediocre food quality.
WITH-REDUCTION	JUSTIFY	Penang has the best overall quality among the selected restaurants.	Penang has very good decor.	Penang has the best overall quality among the selected restaurants, with very good decor.
RELATIVE-CLAUSE	JUSTIFY	Baluchi's has the best overall quality among the selected restaurants.	Baluchi's is located in uptown Manhattan.	Baluchi's, which is located in uptown Manhattan, has the best overall quality among the selected restaurants.
CUE-WORD-CONJUNCTION (<i>because</i>)	JUSTIFY	Caffe Cielo, which is an Italian restaurant, has good service, with good decor.	Caffe Cielo has the best overall quality among the selected restaurants.	Because Caffe Cielo, which is an Italian restaurant, has good service, with good decor, it has the best overall quality among the selected restaurants.
CUE-WORD-CONJUNCTION (<i>since</i>)	JUSTIFY	Amy's Bread has the best overall quality among the selected restaurants.	Amy's Bread's price is 12 dollars, and it has decent service, with excellent food quality.	Amy's Bread has the best overall quality among the selected restaurants since it has mediocre decor, its price is 12 dollars, and it has decent service, with excellent food quality.
PERIOD	JUSTIFY	Amy's Bread has decent service and mediocre decor, and its price is 12 dollars.	Amy's Bread has the best overall quality among the selected restaurants.	Amy's Bread has decent service and mediocre decor, and its price is 12 dollars. It has the best overall quality among the selected restaurants.

Figure 5: Examples of clause combining operations

to realize the JUSTIFY relation. Alt 16 is highly rated, with a human rating of 5 and a RankBoost score of .91. However Alt 2 is a poor realization of this plan, with a human rating of 2 and a Rankboost score of .20. Thus the SPG applied to the tp-tree has a strong effect on the quality of the final realization.

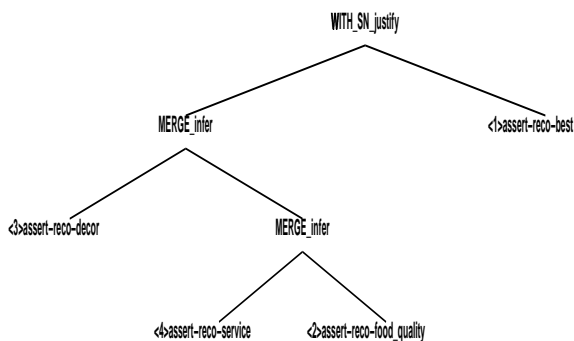


Figure 6: Alternative 16 Sentence Plan Tree

The SPG also handles referring expression generation by converting proper name descriptions of restaurants in an utterance to pronouns when they are also mentioned in the previous utterance. The rules are applied locally, across adjacent sequences of utterances [1]. Referring expressions are manipulated in the DSyntS representations, and the rules apply either (a) during the incremental creation of the sp-tree to utterances conjoined by a CUE-WORD, or (b) after the full sp-tree has been created to all adjacent utterances joined together by a PERIOD. The CUE-WORD CONJUNCTION and PERIOD examples in Figure 5 illustrate the conversion of a named restaurant in the second argument (column 4) to a pronoun (in column 5).

4. Training the Sentence-Plan-Ranker

Examples and Feedback: To apply RankBoost to training the SPR, a set of human-rated sp-trees were encoded in terms of a set of features. We started with a set of 30 text plans for recommendations selected to be representative of SPUR's generation capability. We ran the SPG, letting it produce as many as 20 distinct sp-trees for each text plan. The resulting 600 sentence plans, realized by RealPro, were then rated by two expert judges on a scale from 1 to 5, and the ratings averaged. The ratings assigned to the sp-trees had a mean of 3.9 and a median of 4. Each sp-tree was an example input for RankBoost, with each corresponding rating its feedback.

Features used by RankBoost: RankBoost requires each example to be encoded as a set of real-valued features (binary features are modeled with values 0 and 1). A strength of RankBoost is that the set of features can be very large. We used 7024 features for training the SPR. These features count the number

of occurrences of certain structural configurations, in order to capture decisions made by the randomized SPG in a declarative way. The features are based on feature templates and were automatically generated (using the templates) from the set of sp-trees and associated DSyntS trees that the SPG generated. For this experiment, we distinguish two classes of feature: (1) **Rule-features:** These features are derived from the sp-trees and represent the way in which merge, infer and cue phrase assignment operations are applied to the content plan. These feature names are prefixed with "rule". (2) **Sent-features:** These features are derived from the DSyntSs associated with the nodes of the sp-trees. They describe the deep-syntactic structure of the utterance, including the chosen lexemes. As a result, they are domain specific. These feature names are prefixed with "sent".

We now describe the feature templates used in the discovery process. Three of these templates were used for both sentence plan features and DSyntS features; two were used only for sentence plan features. We distinguish between *local feature templates* which record structural configurations local to a particular node (e.g. its ancestors, daughters etc.), and *global feature templates* which are used only for sentence plan features and record properties of the entire sp-tree. There are four types of local feature template. All local feature templates are instantiated for all nodes in a sp-tree or in a DSyntS tree (except that the LEAF feature is not instantiated in DSyntS trees); the value of the resulting feature is the number of times the described configuration is found in the sp-tree or the DSyntS tree. In all cases, we avoid features specific to particular sentence plans by discarding those that occur fewer than 10 times.

The local feature templates include: traversal features, sister features, ancestor features and leaf features. For each node in the tree, **traversal features** are generated that record the pre-order traversal of the subtree rooted at that node, for all subtrees of all depths (starting with a single-node traversal which just looks at the current node, 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. An example is the feature "rule_traversal_assert-reco-decor" (with value 1) of the left-most leaf of the tree in Figure 6.

Sister features record all consecutive sister nodes. Names are constructed with the prefix "sisters_", followed by the concatenated names of the sister nodes. An example is the feature "rule_sisters_MERGE_infer*assert-reco-best" (with value 1) of the tree in Figure 6.

For each node in the tree, **ancestor 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). An example is the feature "rule_ancestor_assert-reco-decor*MERGE_infer" (with value 1) of the tree in Figure 6.

Leaf features record all initial substrings of the frontier of the sp-tree. Names are prefixed with "leaf_", followed by the

concatenated names of the frontier nodes (starting with the current node). These are binary features. For example, the sentence plan tree of Figure 6 has value 1 for the feature “leaf_#assert-reco-decor#assert-reco-service”.

Global features apply only to the sp-tree. They record, for each sp-tree and for each operation labeling a non-frontier node, (1) the minimal number of leaves 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 in Figure 6 has value 3 for “MERGE_infer_max”, value 2 for “MERGE_infer_min” and value 2.5 for “MERGE_infer_avg”.

5. Experimental Results

Using 2-fold cross validation, we repeatedly tested SPaRky on the half of the corpus of 600 sp-trees that were held out as test data for each fold. The evaluation metric is the human-assigned score for the variant that was rated highest by SPaRky for each content plan for each task/user combination. We evaluated SPaRky on the test sets by comparing for each content plan:

- HUMAN: The score of the top-ranked sentence plan.
- SPARKY: The score of the SPR’s selected sentence.
- RANDOM: The score of a sentence plan randomly selected from the alternate sentence plans.

System	Min	Max	Mean	S.D.
SPaRky	2	5	3.6	.71
HUMAN	2.5	5	3.9	.55
Random	1.5	5	2.9	.88

Table 1: Summary of Recommendation Results (N = 60)

Table 1 summarizes the differences between SPaRky, Human and Random. The mean score of the best sentence plans selected by human judges is 3.9, the mean for SPaRky is 3.6 and the mean for Random selection is 2.9. A statistical comparison of Human, SPaRky and Random with paired t-tests showed that SPaRky was significantly better than Random (df = 59, t = 5.6, p < .001), and significantly worse than Human (df=59, t = 5.9, p < .001).

Table 2 shows some rules that RankBoost learned for ranking input sp-trees from the first test fold. We selected a subset of rules that apply to Alt 12 and Alt 16 from Figure 2. Several rules pertain to the linear order of the resulting output. For example, the feature “leaf_#assert-reco-best” describes a tree configuration in which the speech act “assert-reco-best” is the leftmost leaf in the tree. Thus Rule 1 says that the score of any tree with that speech-act as the leftmost is increased by 0.500, and if this speech act is followed by “assert-reco-food-quality” (Rule 2) then the score goes up by an additional 0.145. Several of the learned rules are related to the amount and type of aggregation in the SPG. For example, Rules 12 and 13 state that if there are 3 or more PROPERNOUN_RESTAURANT nodes in the tree, then decrease the score. The occurrence of so many proper nouns indicates that clause-combining operations that identify identical subjects and combine their attributions were not applied. Rules 4, 5, and 9 indicate that particular types of clause-combining operations are preferred.

6. Discussion

This paper describes SPaRky, a stochastic sentence planner that handles input conceptual structures for information presentation that are more complex and require more attention to semantic

N	Condition	α_s
1	leaf_#assert-reco-best > -∞	0.500
2	leaf_#assert-reco-best#assert-reco-food_quality > -∞	0.145
3	sent_sisters_PROPERNOUN_RESTAURANT_ILquality_ATTR_AMONG_ATTR_with > -∞	0.137
4	sent_traversal_COORD_AND2 ≥ 1.50	0.0434
5	rule_traversal_MERGE_infer ≥ -∞	0.031
6	rule_sisters_assert-reco_CW_CONJUNCTION_infer ≥ 2.50	0.076
7	sent_sisters_ILquality > -∞	0.026
8	sent_sisters_ILHAVE1 ≥ 2.50	0.021
9	sent_traversal_depth0_COORD_AND2 ≥ 1.50	0.020
10	sent_traversal_depth0_HAVE1 ≥ 2.50	-0.339
11	sent_traversal_ILdecor > -∞	-0.167
12	sent_traversal_depth0_PROPERNOUN_RESTAURANT ≥ 2.50	-0.152
13	sent_traversal_PROPERNOUN_RESTAURANT ≥ 2.50	-0.079
14	sent_traversal_depth0_ILdecor > -∞	-0.058
15	sent_sisters_PROPERNOUN_RESTAURANT_ILdecor ≥ -∞	-0.058
16	leaf_count_assert-reco-decor ≥ -∞	-0.041
17	rule_traversal_assert-reco ≥ 4.50	-0.0292
18	rule_traversal_depth0_assert-reco ≥ 4.50	-0.028
19	sent_traversal_depth0_ATTR_with ≥ 1.50	-0.020

Table 2: The rules generated on the first test fold which have the largest impact on the final RankBoost score for recommend Alt 12 and Alt 16. α_s represents the increment or decrement associated with satisfying the condition.

constraints than those used in an SPG for information gathering. Our results show: (1) the method used to train SPoT can be applied in a new domain and for information presentation; (2) that the sentence plans that SPaRky generates are of fairly high quality; and (3) that the training method learns effectively from the data to discriminate high quality sentence plans. In future work, we hope to extend SPaRky to use additional linguistic constructs and apply it to information presentation in a new domain.

7. References

- [1] S. E. Brennan, M. W. Friedman, and C. J. Pollard. A centering approach to pronouns. In *Proc. 25th Annual Meeting of the ACL, Stanford*, pp. 155–162, 1987.
- [2] E. H. Hovy. Automated discourse generation using discourse structure relations. *Artificial Intelligence Journal*, 63:341–385, 1993.
- [3] M. Johnston, S. Bangalore, G. Vasireddy, A. Stent, P. Ehlen, M. Walker, S. Whittaker, and P. Maloor. MATCH: An architecture for multimodal dialogue systems. In *Annual Meeting of the Association for Computational Linguistics, ACL, 2002*.
- [4] N. Karamanis and H. M. Manurung. Stochastic text structuring using the principle of continuity. In *Proc. of INLG-02*, pp. 81–88, 2002.
- [5] B. Lavoie and O. Rambow. A fast and portable realizer for text generation systems. In *Proc. of the Third Conference on Applied Natural Language Processing, ANLP97*, pp. 265–268, 1997.
- [6] W.C. Mann and S.A. Thompson. Rhetorical structure theory: Description and construction of text structures. In Gerard Kempen, editor, *Natural Language Generation*, pp. 83–96, 1987.
- [7] C. Mellish, A. Knott, J. Oberlander, and M. O’Donnell. Experiments using stochastic search for text planning. In *Proc. of International Conference on Natural Language Generation*, pp. 97–108, 1998.
- [8] R. E. Schapire. A brief introduction to boosting. In *Proc. of the 16th International Joint Conference on Artificial Intelligence*, 1999.
- [9] M. A. Walker, S. J. Whittaker, A. Stent, P. Maloor, J. D. Moore, M. Johnston, and G. Vasireddy. Speech-Plans: Generating evaluative responses in spoken dialogue. In *In Proc. of INLG-02.*, 2002.
- [10] M. Walker and O. Rambow, editors. *Computer Speech and Language: Special Issue on Spoken Language Generation*, volume 16: 3-4, 2002.
- [11] M. Walker, O. Rambow, and M. Rogati. Training a sentence planner for spoken dialogue using boosting. *Computer Speech and Language: Special Issue on Spoken Language Generation*, 2002.
- [12] S. J. Young. Talking to machines (statistically speaking). In *IC-SLP*, 2002.
- [13] S. J. Young and F. Fallside. Speech synthesis from concept: a method for speech output from information systems. *Journal of the Acoustic Society of America*, 66(3):685–695, 1979.