Generating Individualized Utterances for Dialogue Systems

François Mairesse
Department of Computer Science
University of Sheffield
Sheffield, S1 4DP, United Kingdom
F.Mairesse@sheffield.ac.uk

Marilyn Walker
Department of Computer Science
University of Sheffield
Sheffield, S1 4DP, United Kingdom
M.A.Walker@sheffield.ac.uk

Abstract

One of the most robust findings of studies of human-human dialogue is that people adapt their utterances to their conversational partners. However, spoken language generators are limited in their ability to adapt to individual users. While statistical models of language generation have the potential for individual adaptation, we know of no experiments showing this. In this paper, we utilize one statistical method, boosting, to train a spoken language generator for individual users. We show that individualized models perform better than models based on sets of users, and describe differences in the learned individual models arising from the linguistic preferences of users.

1 Introduction

One of the most robust findings of studies of human-human dialogue is that people adapt their utterances to their conversational partners, according to perceived individual differences among addressees, e.g. age, dialect, social group, or intelligence, and that this adaptation affects most aspects of utterance production [1, 2]. Experiments have also shown that users will adapt to dialogue systems and that they prefer systems whose linguistic expression of personality matches the user's [3, 4]. However, spoken language generators (SLGs) are limited in their ability to adapt to individual users. There are successful methods for user-specific content selection that result in higher user satisfaction [5], and techniques for adapting the speaking rate to the user [6], but many sources of variation are not modelled. This limitation may arise from the highly handcrafted nature of most SLGs, making it impossible to target individual users. Statistical models of language generation have the potential for individual adaptation given appropriate training data [7, 8], but we know of no experiments showing this.

This paper examines the feasibility and utility of training a SLG for individual users, using the SPaRKy generator [5]. SPaRKy generates spoken utterances in the restaurant domain in

relations:	justify(nuc:1, sat:2); justify (nuc:1, sat:3); justify(nuc:1, sat:4); justify(nuc:1, sat:5)
content:	1. assert(best(Chanpen Thai)) 2. assert(is(Chanpen Thai, cuisine(Thai))) 3. assert(has(Chanpen Thai, food-quality(good))) 4. assert(has(Chanpen Thai, service(good))) 5. assert(is(Chanpen Thai, price(24 dollars)))

Figure 1: Content Plan for a recommendation.

response to requests from the user to *recommend* a restaurant or to *compare* sets of restaurants. The input to SPaRKy is a *content plan*, which is a set of *assertions* about restaurants that the user is considering, and a specification of the *rhetorical relations* that hold between those facts, as shown in Fig. 1. The relations in Fig. 1 specify that assertion (1) is the *claim* being made in the recommendation, and that the other assertions (2 to 5) provide justifying *evidence* for the claim.

SPaRKy can be used to train an individualized SLG because the Sentence Plan Generator

Alt	Realization	A	B	SPR_A	SPR_B
6	Chanpen Thai has the best overall quality among the selected restaurants since it is a Thai restaurant, with good service, its price is 24 dollars, and it has good food quality.	1	4	0.16	0.65
9	Chanpen Thai is a Thai restaurant, with good food quality, its price is 24 dollars, and it has good service. It has the best overall quality among the selected restaurants.	2	4	0.47	0.53
1	Chanpen Thai has the best overall quality among the selected restaurants. This Thai restaurant has good food quality. Its price is 24 dollars, and it has good service.	4	3	0.64	0.52
8	Chanpen Thai is a Thai restaurant, with good food quality. It has good service. Its price is 24 dollars. It has the best overall quality among the selected restaurants.	4	2	0.81	0.29

Figure 2: Some alternative outputs for the content plan in Fig. 1, with feedback from Users A and B (1=worst and 5=best) and rankings from the trained individual SPRs ($[0, 1]$).

(SPG) (Section 2.1) generates competing outputs and the Sentence Plan Ranker (SPR) (Section 2.3) is trained from user feedback. The training method requires competing outputs to be rated by the target user. Linguistically motivated features that describe these examples are automatically discovered (Section 2.2). Then the SPR is trained, on the basis of the featural representation of each example and its feedback, to duplicate the rankings in the training examples. To do this, it produces a rule-based model of the effect of each feature on the ranking of the competing examples. The models allow us to compare the effects of the features on the rankings in order to understand the preferences of individual users.

Operation	**Relation**	**Description**	**Sample 1st arg**	**Sample 2nd arg**	**Result**
MERGE	INFER	Two clauses can be combined if they have identical matrix verbs and identical arguments and adjuncts except one. The non-identical arguments are coordinated.	Chanpen Thai has good service.	Chanpen Thai has good food quality.	Chanpen Thai has good service and good food quality.
WITH-REDUCTION	JUSTIFY or INFER	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.	Chanpen Thai is a Thai restaurant.	Chanpen Thai has good food quality.	Chanpen Thai is a Thai restaurant, with good food quality.
RELATIVE-CLAUSE	JUSTIFY or INFER	Two clauses with an identical subject can be identified. One clause is attached to the subject of the other clause as a relative clause.	Chanpen Thai has the best overall quality among the selected restaurants.	Chanpen Thai is located in Midtown West.	Chanpen Thai, which is located in Midtown West, has the best overall quality among the selected restaurants.
CUE-WORD-CONJUNCTION (*because*, *since*, *and*)	JUSTIFY	Two clauses are conjoined with a cue word (coordinating or subordinating conjunction).	Chanpen Thai has the best overall quality among the selected restaurants.	Chanpen Thai is a Thai restaurant, with good service.	Chanpen Thai has the best overall quality among the selected restaurants, since it is a Thai restaurant, with good service.
PERIOD	JUSTIFY	Two clauses are joined by a period.	Chanpen Thai is a Thai restaurant, with good food quality.	Chanpen Thai has good service.	Chanpen Thai is a Thai restaurant, with good food quality. It has good service.

Figure 3: Clause combining operations and examples.

Why do we believe that there will be individual differences in the learned user models? Fig. 2 shows some competing outputs for the plan in Fig. 1. The rating feedback from two users are in columns A and B. Inspection of the ratings suggests that each user has different perceptions as to the quality of the outputs. These different perceptions are not unique to this one example. A paired t-test on the feedback from Users A and B, for 600 examples, shows that the feedback of the two users represent independent populations ($t = 17.38$, $p < 0.001$). A correlation analysis of the 20 realizations for each of the 30 content plans gives a mean Kendall's τ value of 0.039, which indicates that on average the ratings aren't correlated at all. Moreover, we also examined the user feedback from an experiment where 60 users rated the output of 7 different spoken language generators for

20 content plans, and again found significant differences in user perceptions of utterance quality ($F = 1.2$, $p < 0.002$) [10]. Thus, these data indicate that there is a potentially high utility for training individualized SPRs. This paper describes experiments with an individualized generator for recommendations.

2 The SPaRKy Trainable Generator

2.1 Sentence Plan Generation

The SPG first hierarchically groups the assertions in the input content plan that talk about the same thing, leaving their linear order unspecified. The result is one or more text-plan trees (**tp-trees**), with leaves labelled by speech acts and interior nodes labelled by rhetorical relations. Each leaf of the tp-tree is associated with one or more syntactic means of realizing these assertions, using a dependency tree representation [11]. Then, the SPG applies the clause-combining operations in Fig. 3 to the set of tp-trees, two nodes at a time. The result is two parallel structures: (1) the **sp-tree**, a binary tree with leaves labeled by the speech acts from the input content plan, and interior nodes labeled with clause-combining operations; and (2) the **d-tree** which reflect the parallel operations on the dependency tree representations. To generate a random sample of competing **sp-trees** the operations are randomly selected, but restricted by the rhetorical relations holding at the interior nodes. **D-trees** are passed to the RealPro surface realizer for the final generation step [11].

The **sp-trees** for Alts 6 and 8 in Fig. 2 are in Figs. 4 and 5. Leaf labels are concise names for assertions in the content plan, e.g. **assert-reco-best** is the claim (labelled 1) in Fig. 1. Because combination operations can switch the order of their arguments, from satellite before nucleus (SN) to nucleus before satellite (NS), the labels on the interior nodes indicate whether this occurred, and specify the rhetorical relation that the operation realizes. For example, the label at the root of the tree in Fig. 4 (**CW-SINCE-NS-justify**) specifies that the CW-CONJUNCTION operation was used, with the *since* cue word, with the nucleus first (NS), to realize the *justify* relation.

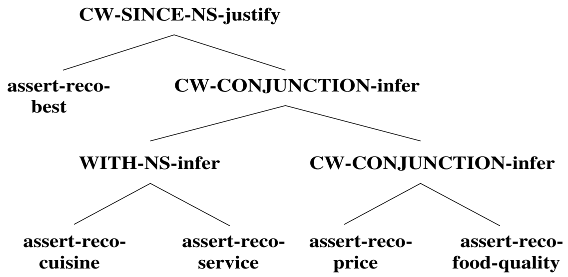

Figure 4: Alternative 6 Sentence Plan Tree.

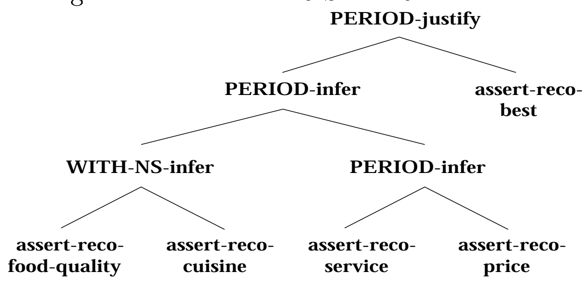

Figure 5: Alternative 8 Sentence Plan Tree.

2.2 Feature Generation

Each example output of the SPG is represented by a set of features. The features are based on feature templates and are automatically generated from the sp-trees and their associated d-trees.

The features describe decisions made by the SPG by counting the occurrences of certain structural configurations. We use the same feature discovery algorithm as the one described in [5]. **Rule-features** are derived from the sp-trees and represent how the combination operations were applied to the tp-tree. These feature names start with "R-". **Sent-features** are derived from the d-trees associated with each node of the sp-trees, including the chosen lexemes. These feature names start with "S-". For each type of tree, for each node in the tree:

- **Traversal features** record the preorder traversal of the subtree rooted at that node, for all subtrees of all depths. An example is R-TRAVERSAL-WITH-NS-INFER*ASSERT-RECO-FOOD-QUALITY*ASSERT-RECO-CUISINE (with value 1) of the bottom-left subtree in Fig. 5.
- **Sister features** record consecutive sister nodes. An example is R-SISTER-ASSERT-RECO-BEST*CW-CONJUNCTION-INFER (with value 1) of the tree in Fig. 4.
- **Ancestor features** record all the initial subpaths of the path from that node to the root. An example is R-ANCESTOR-ASSERT-RECO-CUISINE*WITH-NS-INFER*CW-CONJUNCTION-INFER (with value 1) of the tree in Fig. 4.
- **Leaf features** record all initial substrings of the frontier of the sp-tree. These are binary features. For example, the sp-tree of Fig. 4 has value 1 for LEAF-ASSERT-RECO-BEST and also for LEAF-ASSERT-RECO-BEST*LEAF-ASSERT-RECO-CUISINE, and the sp-tree of Fig. 5 has value 1 for LEAF-ASSERT-RECO-FOOD-QUALITY*ASSERT-RECO-CUISINE
- **Global features** record, for each sp-tree and for each operation labeling a non-frontier node, the (1) minimal, (2) maximal and (3) average number of leaves dominated by a node labeled with that rule in that tree. For example, the sp-tree in Fig. 4 has value 4 for CW-CONJUNCTION-INFER-MAX, value 2 for CW-CONJUNCTION-INFER-MIN and value 3 for CW-CONJUNCTION-INFER-AVG.

Both 'lexicalized' and 'nonlexicalized' of all features are generated using these templates, then those occurring less than 10 times overall are discarded, resulting in 7024 unique features.

2.3 Training an Individualized Sentence Plan Ranker

The training method utilizes the RankBoost algorithm [9]. 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 2.2. For example, one indicator function is:

$$h_{100}(x) = \begin{cases} 1 & \text{if } \textsc{leaf-assert-reco-best}(x) \geq 1 \\ 0 & \text{otherwise} \end{cases}$$

So $h_{100}(x) = 1$ if the leftmost leaf is the assertion of the claim as in Fig. 4. Each indicator function has a parameter α_s, and the 'ranking score' for an example x is calculated as:

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

This score is used to rank competing sp-trees of the same content plan to reproduce the ranking in the training data. The user ratings are converted into a training set $\mathcal{T}$ of *ordered pairs* of examples x, y:

$$\mathcal{T} = \{(x,y) \mid x, y \text{ are alternatives for the same plan,} \\ x \text{ is preferred to } y \text{ by user ratings}\}$$

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

$$Loss = \frac{1}{|\mathcal{T}|} \sum_{(x,y)\in\mathcal{T}} eval(F(x) \leq F(y))$$

The *eval* function returns 1 if the ranking scores of the (x, y) pair are misordered, and 0 otherwise. As this loss function is minimized, the ranking errors are 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. In the experiments below, the value of the loss function is reported as the testing error rate.

An important aspect of RankBoost is that the learned models are expressed as rules, which supports the analysis of differences in the users' models. Each rule modifies the final ranking score F by α_s whenever a feature value is above a particular threshold. For example, here is a rule of User A's model, described later as Rule 18 in Fig. 6:

if R-ANCESTOR-ASSERT-RECO-FOOD-QUALITY*MERGE-INFER ≥ 1 then increase α by 0.18

This rule is triggered whenever the sentence plan tree of the alternative to be ranked includes at least one branch in which a MERGE-INFER node is a direct ancestor of an ASSERT-RECO-FOOD-QUALITY node. If it is the case, the rule will increase the total ranking score by 0.18. Since it improves the rating of the sentences which satisfy its condition, such a rule implies a preference for food quality information to be combined with another assertion using the MERGE operation. Preferences are discussed in more details in section 3.2.

To qualitatively compare the learned ranking models for the individualized SPRs, we assess: (1) Whether or not an individual is oriented towards a particular linguistic aspect of the utterance, by examining which indicator functions $h_s(x)$ (one for each feature) have non-zero values; and (2) How important a feature is to an individual, by examining the magnitude of the parameter α_s. To facilitate these comparisons, we apply a feature selection algorithm that reduces the number of rules to 100 per user.

3 Results

3.1 Quantitative analysis

To quantify the utility and the feasibility of training individualized SLGs, we first investigate the accuracy of the individualized SPRs as tested via 2-fold cross-validation. Table 1 shows the ranking loss for several training and testing configurations (lower is better). We compare the individualized models with models trained on A and B's mean feedback (AVG) [5] on their own test data and on test data for another model. The most striking differences are between models trained and tested on their own test data with error rates around 0.15, and cross model error rates (from 0.26 to 0.51). Note that models trained on the means perform poorly for both users A and B. As a baseline for comparison, a model ranking sentence alternatives randomly produces a ranking loss of 0.5 on average.

Configuration	A's model	B's model	AVG model
A's test data	0.16	0.51	0.31
B's test data	0.51	0.15	0.30
AVG test data	0.27	0.26	0.15

Table 1: Error rates for various learning configurations.

The second issue is the feasibility of training models for individual users. Table 1 is based on a corpus of 600 examples, rated by each user, which may involve too much effort to acquire. However, an analysis of error rates as a function of the amount of training data shows error rates around 0.20 can be achieved with a training set of 120 examples.

3.2 Qualitative analysis

We qualitatively compare the models of Users A and B, by identifying differences among the features selected by RankBoost, and by examining their α values. User A's model consists of 109 rules; a subset are in Fig. 6. User B's model consists of 90 rules, with a subset in Fig. 7. We first consider how these models account for the rating differences for Alt-6 and Alt-8 in Fig. 2, and then discuss other differences. The reader can refer to section 2 for an explanation of the elements of the features used and a description of the structure of the rules. To clarify the general use of the rules in our analysis, Fig. 8 shows how User A's Rule 10 and User B's Rule 22 are activated

N	Condition	α
1	R-ANCESTOR-ASSERT-RECO-NBHD*WITH-NS-INFER $\geq$ 1	-1.26
3	R-ANCESTOR-ASSERT-RECO*WITH-NS-INFER*CW-CONJUNCTION-INFER $\geq$ 1	-0.33
5	CW-CONJUNCTION-INFER-AVG-LEAVES-UNDER $\geq$ 2.8	-0.27
7	R-ANCESTOR-CW-CONJUNCTION-INFER*CW-CONJUNCTION-INFER $\geq$ 1	-0.17
8	WITH-NS-INFER-MIN-LEAVES-UNDER $\geq$ 1	-0.13
9	R-ANCESTOR-ASSERT-RECO*WITH-NS-INFER $\geq$ 1	-0.11
10	CW-CONJUNCTION-INFER-MAX-LEAVES-UNDER $\geq$ 3.5	-0.07
11	R-TRAVERSAL-WITH-NS-INFER*ASSERT-RECO*ASSERT-RECO $\geq$ 1	-0.07
12	R-ANCESTOR-ASSERT*WITH-NS-INFER $\geq$ 1	-0.03
15	CW-CONJUNCTION-INFER-AVG-LEAVES-UNDER $\geq$ 4.1	-0.01
18	R-ANCESTOR-ASSERT-RECO-FOOD-QUALITY*MERGE-INFER $\geq$ 1	0.18
19	R-ANCESTOR-ASSERT-RECO*MERGE-INFER $\geq$ 2.5	0.20
21	R-ANCESTOR-ASSERT*MERGE-INFER $\geq$ 2.5	0.25
22	R-TRAVERSAL-MERGE-INFER $\geq$ 1.5	0.27
23	R-TRAVERSAL-WITH-NS-INFER*ASSERT-RECO-SERVICE*ASSERT-RECO-FOOD-QUALITY $\geq$ 1	0.40
24	LEAF-ASSERT-RECO-FOOD-QUALITY*ASSERT-RECO-CUISINE $\geq$ 1	0.46
26	LEAF-ASSERT-RECO-FOOD-QUALITY $\geq$ 1	0.60

Figure 6: Rules and corresponding α values of User A's model, ordered by α.

N	Condition	α
1	R-SISTER-ASSERT-RECO-RELATIVE-CLAUSE-INFER $\geq$ 1	-1.01
2	R-SISTER-PERIOD-INFER-ASSERT-RECO $\geq$ 1	-0.71
3	R-ANCESTOR-ASSERT-RECO-NBHD*WITH-NS-INFER $\geq$ 1	-0.50
4	R-ANCESTOR-ASSERT-RECO*PERIOD-INFER*PERIOD-INFER $\geq$ 1.5	-0.49
5	R-ANCESTOR-ASSERT-RECO-FOOD-QUALITY*WITH-NS-INFER*RELATIVE-CLAUSE-INFER $\geq$ 1	-0.41
6	R-ANCESTOR-ASSERT-RECO-CUISINE*WITH-NS-INFER*RELATIVE-CLAUSE-INFER $\geq$ 1	-0.39
7	R-ANCESTOR-ASSERT-RECO*PERIOD-INFER $\geq$ 1	-0.35
9	R-ANCESTOR-ASSERT*PERIOD-INFER*PERIOD-INFER $\geq$ 1.5	-0.26
12	R-TRAVERSAL-RELATIVE-CLAUSE-INFER*ASSERT-RECO*WITH-NS-INFER $\geq$ 1	-0.05
18	R-ANCESTOR-ASSERT-RECO-CUISINE*WITH-NS-INFER*CW-CONJUNCTION-INFER $\geq$ 1	0.27
19	CW-CONJUNCTION-INFER-AVG-LEAVES-UNDER $\geq$ 3.6	0.36
20	LEAF-ASSERT-RECO-BEST $\geq$ 1	0.47
21	LEAF-ASSERT-RECO-BEST*ASSERT-RECO-CUISINE $\geq$ 1	0.50
22	CW-CONJUNCTION-INFER-AVG-LEAVES-UNDER $\geq$ 2.8	0.52
23	R-TRAVERSAL-WITH-NS-INFER*ASSERT-RECO-CUISINE*ASSERT-RECO-FOOD-QUALITY $\geq$ 1	0.76

Figure 7: Rules and corresponding α values of User B's model, ordered by α.

based on the sp-tree of Alt-6, and Fig. 9 describes how User A's Rule 9 and User B's Rule 2 both decrease Alt-8's ranking score.

3.2.1 Comparison of alternatives 6 and 8

Alt-6 is highly ranked by User B but not by User A. Alt-6 instantiates Rules 20 and 21 of Fig. 7, expressing User B's preferences about linear order of the content. As mentioned in section 2.2, LEAF features represent the ordering of the assertions starting from the beginning of the sentence. Rule 20 increases the rating of examples in which the claim ASSERT-RECO-BEST (*Chanpen Thai has the best overall quality*) is realized first. The second rule promotes examples in which the initial claim is immediately followed by the type of cuisine (ASSERT-RECO-CUISINE), as illustrated by Fig. 8. One can see that Alt-6's sp-tree satisifies the condition of both rules as the left-most leaf is ASSERT-RECO-BEST, and the second one is ASSERT-RECO-CUISINE. These rules interact with Rule 18 in Fig. 7, which is activated whenever the speech act ASSERT-RECO-CUISINE is a child node of a WITH-NS-INFER, which is itself a child node of a CW-CONJUNCTION-INFER node, as in Fig. 4. This rule specifies a preference for information following ASSERT-RECO-CUISINE to be combined via the WITH-NS operation, and then conjoined (CW-CONJUNCTION-INFER) with additional evidence. This structure corresponds to *"...it is a Thai restaurant, with..."* in Alt-6's realization in Fig. 2. Alt-6 also instantiates Rule 22 in User B's model, with an α value of .52 associated with multiple uses of the CW-CONJUNCTION-INFER operation, i.e. many inferences realized by conjunctions. This feature is realized through the multiple commas and the final *and* conjunction.

User A's low rating of Alt-6 arises from A's dislike of the WITH-NS operation (Rules 3, 8, 9, 11 and 12) and the CW-CONJUNCTION-INFER operation (Rules 3, 5, 7, 10 and 15) in Fig. 6. Rule 10 is explained in details in Fig. 8. Contrast User B's Rule 22 with User A's Rules 5, the first one associates a very high α value whenever the CW-CONJUNCTION-INFER operation combines more than 2.8 leaves on average, while the second one decreases the ranking score by 0.27 given the same feature. Alt-6 also fails to instantiate A's preference for food quality and cuisine information to occur initially (Rules 24 and 26).

Alt-8 is rated highly by User A but not by User B (see Fig. 2). Alt-8 instantiates the negatively

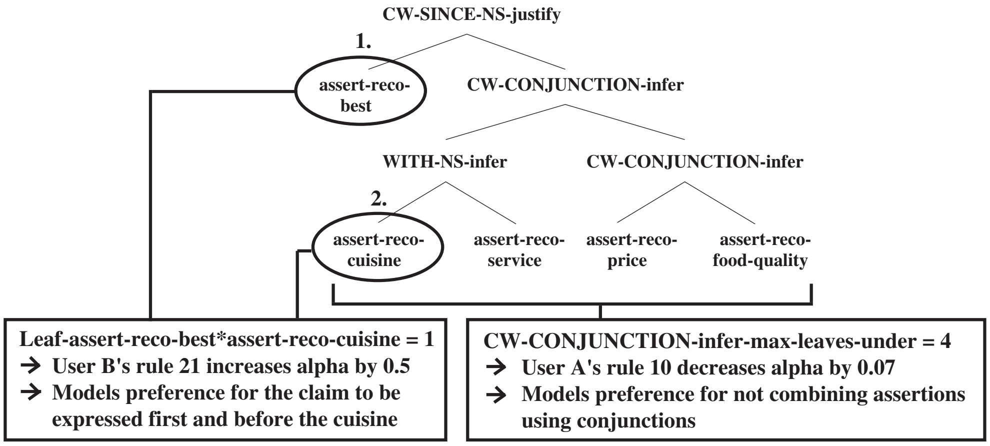

Figure 8: Evaluation of user preferences for the sp-tree of alternative 6, using Rule 10 in Fig. 6 and Rule 21 in Fig. 7. The feature values are first identified, and the corresponding rules are used to compute the total α ranking score.

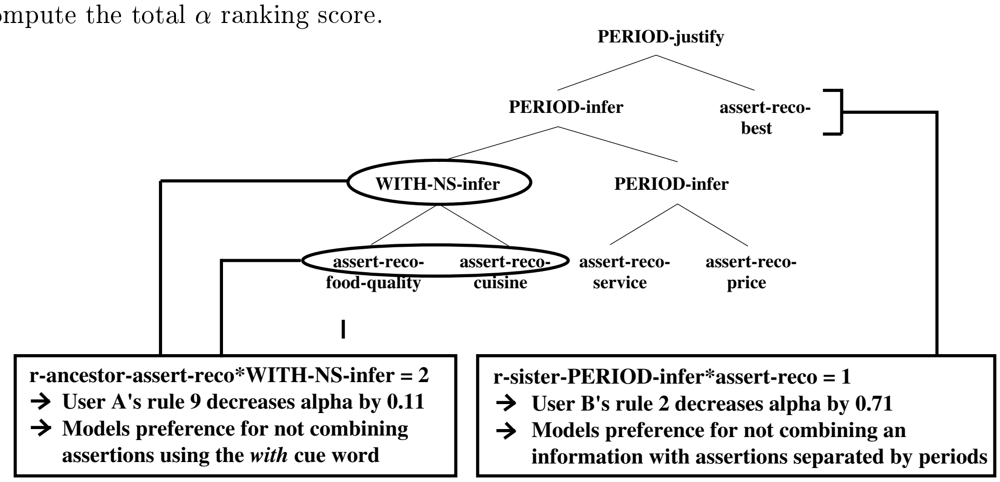

Figure 9: Evaluation of user preferences for the sp-tree of alternative 8, using Rule 9 in Fig. 6 and Rule 2 in Fig. 7.

evaluated WITH-NS operation in User A's model (Rules 8, 9, 11 and 12 in Fig. 6), the application of Rule 9 is detailed in Fig. 9. However, there are no instances of CW-CONJUNCTION-INFER as all inferences are realized either by the *with* cue word or by periods. Therefore, Alt-8's sp-tree in Fig. 5 doesn't activate the highly negative rules 3, 5, 7, 10 and 15. Moreover, Alt-8 follows A's ordering preferences (Rules 24 and 26) which describe sp-trees with ASSERT-RECO-FOOD-QUALITY on the left frontier, and trees where it is followed by ASSERT-RECO-CUISINE[1]. On the other hand, Alt-8 is rated poorly by User B; it violates B's preferences for linear order (Rules 20 and 21) because the claim isn't realized first, and B's model has rules that radically decrease the ranking of examples using the PERIOD-INFER operation (Rules 2, 4, 7 and 9). This operation uses periods to combine inferences, which produces realizations composed of multiple sentences. As an example, the effect of Rule 2 on the evaluation of Alt-8's sp-tree is detailed in Fig. 9.

[1]Here, the WITH-NS rule changes the order of the arguments in the d-tree so it describes the placement of these facts in the initial sentence, and that the main clause is ASSERT-RECO-FOOD-QUALITY.

3.2.2 Other differences

There are also individual preferences for particular operations in general and for specific content operation interactions. For example, User A's model demotes examples where the WITH-NS operation has been applied, and more so when expressing neighborhood information (Rule 1), but User A likes WITH-NS when it combines cuisine and food-quality information (Rule 23). User A also likes the MERGE-INFER operation (Rules 19, 21 and 22), which merges two assertions with identical subjects to produce one sentence. This preference is increased when one of the assertions combined is ASSERT-RECO-FOOD-QUALITY (Rule 18). In contrast, User B likes the CW-CONJUNCTION operation (Rule 19 in Fig. 7), and doesn't like the WITH-NS operation for combining neighborhood information (Rule 3). User B also dislikes the RELATIVE-CLAUSE-INFER operation in general (Rule 1), and its combination with the WITH-NS operation (Rule 12). An interesting interaction arises from Rules 5, 6 and 23. User B strongly prefers the WITH-NS operation for combining cuisine and food-quality information (Rule 23), but radically reduces the rating of this sp-tree configuration when it is combined with further information using the RELATIVE-CLAUSE-INFER operation (Rules 5 and 6).

4 Conclusion

This paper presents a statistical method for training individualized spoken language generators. While it seems clear that modeling individual differences is important for spoken language generation, we know of no other similar work. We show that users have different perceptions as to the quality of different system responses, and that individualized models perform better than those trained for sets of users. We also discuss qualitative differences in the models of two users, showing that these differences can be attributed to individual linguistic preferences in sentence planning. An interesting extension of the results presented here would be a system that could learn an addressee's model in real-time during a dialogue.

References

[1] S. E. Brennan. Lexical entrainment in spontaneous dialog. In *Proc. of 1996 International Symposium on Spoken Dialogue*, p. 41–44, 1996.

[2] S. Garrod and A. Anderson. Saying what you mean in dialogue: A study in conceptual and semantic coordination. *Cognition*, 27:181–218, 1987.

[3] B. Reeves and C. Nass. *The Media Equation*. University of Chicago Press, 1996.

[4] C. Darves and S. Oviatt. Adaptation of Users' Spoken Dialogue Patterns in a Conversational Interface. In *Proc. of 7th ICSLP*, p. 561–564, 2002.

[5] M. Walker, R. Prasad and A. Stent. A trainable generator for recommendations in multimodal dialogue. In *Proc. of Eurospeech 2003*, p. 1697–1701, 2003.

[6] N. Ward and S. Nakagawa. Automatic user-adaptive speaking rate selection for information delivery. In *Proc. of 7th ICSLP*, p. 549–552, 2002.

[7] A. Oh and A. Rudnicky. Stochastic natural language generation for spoken dialog systems. *Computer Speech and Language*, 16(3-4):387–407, 2002.

[8] A. Ratnaparkhi. Trainable approaches to surface natural language generation and their application to conversational dialog systems. *Computer Speech and Language*, 16(3-4):435–455, 2002.

[9] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. In *Proc. of 15th ICML*, p. 170–178, 1998.

[10] O. Rambow, M. Rogati, M. Walker. Evaluating a trainable sentence planner for a spoken dialogue system. In *Proc. of Annual Meetings of ACL*, p. 426–433, 2001.

[11] B. Lavoie and O. Rambow. A fast and portable realizer for text generation systems. In *Proc. of 3rd Conference on ANLP*, p. 265–268, 1997.