Can We Talk? Methods for Evaluation and Training of Spoken Dialogue Systems

Marilyn A. Walker
M.A.Walker@sheffield.ac.uk
University of Sheffield

Abstract.
There is a strong relationship between evaluation and methods for automatically training language processing systems, where generally the same resource and metrics are used both to train system components and to evaluate them. To date, in dialogue systems research, this general methodology is not typically applied to the dialogue manager and spoken language generator. However, any metric for evaluating system performance can be used as a feedback function for automatically training the system. This approach is motivated with examples of the application of reinforcement learning to dialogue manager optimization, and the use of boosting to train the spoken language generator.

Keywords: dialogue systems, machine learning, evaluation

1. Introduction

It is obvious that there is a strong relationship between evaluating a system and automatically training it: the relationship arises from the fact that the same data can be used to train a system and to evaluate it. Examples from language technology include speech recognition, where the training data consists of a speech corpus and the corresponding human transcriptions, and evaluation compares the recognizer output with the transcription, and part of speech tagging and parsing, where the training data consists of a corpus of word strings labeled with part-of-speech tags or parser structures, and evaluation compares the labels with the output of an automatic tagger or parser. There are many other examples.

However, while early work on dialogue system evaluation compared system utterances to those collected in another dialogue (Hirschman, 2000), dialogue systems cannot be properly evaluated in this way.[1] This is because several dialogue strategies may be appropriate at any point in the dialogue, and even within a single strategy there are many choices for both the content and the realization of that strategy. Consider for example the alternative realizations of the same recommendation strategy in Figure 1. The A and B columns are the ratings that Users A

[1] These evaluations were nevertheless valuable in moving the field forward.

Alt	Realization	A	B	RB_A	RB_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 1. Some Alternative Realizations for a recommendation, with feedback from Users A and B and ranking values (RB) for the trained rankers for Users A and B.

and B give to different realizations. The ratings indicate that multiple realizations can be given the same score, and moreover that two users may have different perceptions of the quality of the alternatives (other columns are explained below). Thus, the most that can be said is that some strategies or realizations will be better than others, for particular users, given particular dialogue contexts.

In fact, this is true for every problem in language processing involving the generation of linguistic output. For example, in text-to-text generation, the input structure consists of one or more documents and the output is a summary, or an answer to a question; when users are asked to evaluate different orderings of propositions in the output text, there are many possible acceptable orderings (Barzilay et al., 2002; Lapata, 2003). When generating weather reports from raw measurement data, different expert meteorologists choose different lexical items and syntactic realizations for the same underlying concepts (Reiter, 2002). Similarly, when judging different referring expressions expressing the same concept, users do not agree on the best referring expression for a particular context, i.e. multiple possibilities are acceptable (Yeh and Mellish, 1997).

Nevertheless, the same data can be used to train a dialogue system and to evaluate it; the key is that different evaluation and training methods are required. Our research has experimented with (1) automatically training the dialogue manager, by applying reinforcement learning (Walker, 2000; Litman et al., 2000); and (2) automatically training the spoken language generator using the Rankboost algorithm, a form of boosting (Rambow et al., 2001; Stent et al., 2004; Freund et al., 1998). Both of these algorithms utilize a metric to be optimized;

this metric is called an objective function. Research on dialogue systems offers many possible objective functions typically used for evaluation: (1) Task completion or transaction success ; (2) Efficiency measures such as time to completion; (3) Subjective measures derived from direct user feedback, such as user satisfaction based on a user-survey, or user ratings of system utterances in context. These measures tend to be task-oriented, but measures oriented towards other aspects of the system, such as social engagement (André et al., 2000), are possible. The key aspect of these training algorithms is that they provide a way to associate scalar evaluation metrics with different output choices that a system can make in different contexts, thus inducing a *ranking* over the different choices.

In the remainder of the paper, we discuss methods for training dialogue systems in more detail. Section 2 summarizes experiments using reinforcement learning and Section 3 summarizes the application of boosting. Section **??** discusses implications for future work on the evaluation and optimization of dialogue systems.

2. Applications of Reinforcement Learning for Dialogue Management

We conducted experiments on automatically optimizing the dialogue manager using reinforcement learning in two domains, and for different dialogue strategies. The ELVIS system is a spoken dialogue interface to email (Walker, 2000), and the NJFun system provides information about fun things to do in New Jersey (Litman et al., 2000). Applying reinforcement learning to dialogue relies on three fundamental assumptions:

1. The system is characterized in terms of a set of states, and actions that can be taken in each state;

2. Actions can be something said to the user, a whole subdialogue, or accessing the database;

3. The rewards received on reaching a state or at the end of dialogue are used to learn which actions lead to highest rewards.

System actions are thought of as dialogue strategies, which can be for initiative management, information gathering or presentation, error prevention or recovery, and database query. To train the system, an objective function P_i is computed for each dialogue D_i, and used as the utility of the final state of the dialogue D_i (Sutton and Barto,

1998). Then, the utility of doing action a in state S_i, $U(a, S_i)$ (its Q-value), can be calculated in terms of the utility of a successor state S_j, by obeying the recursive equation:

$$U(a, S_i) = R(a, S_i) + \sum_j M_{ij}^a \max_{a'} U(a', S_j)$$

where $R(a, S_i)$ is the immediate reward received for doing action a in S_i, a is a strategy from a finite set of strategies A that are admissable in state S_i, and M_{ij}^a is the probability of reaching state S_j if strategy a is selected in state S_i. In other words, the utility of strategy a in a state s_i is a function of the reward for getting to that state, and the weighted sum of the rewards for states you can get to from there. In both sets of experiments, the reward associated with each state, $R(S_i)$, is zero. In addition, since a priori prediction of a user action in a particular state is not possible, the state transition model M_{ij}^a is estimated from the logged state-strategy history for the dialogue.

The utility values are then estimated to within a desired threshold using value iteration, which updates the estimate of $U(a, S_i)$, based on updated utility estimates for neighboring states, so that the equation above becomes:

$$U_{n+1}(a, S_i) = R(S_i) + \sum_j M_{ij}^a \max_{a'} U_n(a', S_j)$$

where $U_n(a, S_i)$ is the utility estimate for doing a in state S_i after n iterations (Sutton and Barto, 1998) pp. 101. Value iteration stops when the difference between $U_n(a, S_i)$ and $U_{n+1}(a, S_i)$ is below a threshold, and utility values have been associated with states where strategy selections were made. Selecting the strategy with the maximal Q-value in each dialogue state should lead to optimal system behavior.

The strength of this approach is that the reward function can be any evaluation metric, and the system will optimize its behavior to maximize that reward function. The NJFun system used an objective metric, task completion, whereas the ELVIS email reader used user satisfaction. To train a system that minimizes cognitive load, you simply need a way to measure it and then use that as the reward function.

2.1. The ELVIS System

The experiments with the ELVIS system were the first test of the reinforcement learning approach using interaction with human users for training and for providing feedback. ELVIS provides realtime spoken language access to email from any phone. It can summarize the user's folders by sender, or by subject, and select and read messages from

specific senders or on specific subjects. It can respond to requests for help, to undo or repeat a previous action, and it has a timing facility that detects when the user fails to speak and offers context-specific help. ELVIS also supports *barge-in* so that it can be interrupted at any time, which is vitally important for an email application.

Strategy	Example Realization
Sys-Init	Hi, ELVIS here. You have 5 new or unread messages in your inbox. Say Read, Summarize, or say Help for more options.
Mix-Init	Hi, ELVIS here. I've got your mail.
Summ-Both	In your top level inbox, from Kim, there's 1 message about "Lunch." From Michael, there's 1 message about "Evaluation group meeting." From Noah, there's 1 message about "Call Me Tomorrow" and 1 message about "Interviewing Antonio." And from Owen, there's 1 message about "Agent Personality."
Summ-System	In your top level inbox, there's 1 message from Kim, 2 messages from Noah, 1 message from Michael, and 1 message from Owen.
Summ-Choice	E: Summarize by subject, by sender, or both? U: Subject. E: In your top level inbox, there's 1 message about "Lunch," 1 message about "Interviewing Antonio," 1 message about "Call Me Tomorrow," 1 message about "Evaluation Group Meeting," and 1 message about "Agent Personality."
Read-First	In the messages from Kim, there's 1 message about "Interviewing Antonio" and 1 message about "Meeting Today." The first message is titled, "Interviewing Antonio." It says, "I'd like to interview him. I could also go along to lunch. Kim."
Read-Summ-Only	In the messages from Kim, there's 1 message about "Interviewing Antonio" and 1 message about "Meeting Today."
Read-Choice	In the messages from Kim, there's 1 message about "Interviewing Antonio" and 1 message about "Meeting Today." To hear the messages, say, "Interviewing Antonio" or "Meeting."

Figure 2. Alternate Strategies for Initiative, Summarizing and Reading messages

The training experiments contrasted strategies for mixed or system initiative, and for summarizing folders and presenting sets of email messages. See Figure 2. For training, 78 users interacted with ELVIS to complete three tasks related to accessing information from email messages in their inbox. For testing, six users performed the same three tasks. The reward function was an approximation of user satisfaction predicted from automatically logged variables (Walker et al., 1997). This reward function was appropriate given ELVIS's high level of task success.

S1: Welcome to NJFun. How may I help you?
U1: I'd like to find um winetasting in Lambertville in the morning.
S2: Did you say you are interested in Lambertville?
U2: Yes.
S3: Did you say you want to go in the morning?
U3: Yes.
S4: I found a winery near Lambertville that is open in the morning. It is the Poor Richard's Winery in Lambertville. Thank you for using the system. Please give me feedback by saying 'good', 'so-so', or 'bad'.
U4: Good.
S5: Goodbye and have a nice day!

Figure 3. An example dialogue with NJFun.

The system learned to use the Sys-Init and the Read-First strategy, and to use the Summ-Both strategy at the beginning of the dialogue, and then to switch to using Summ-System. This gives the user complete information about all the messages in her inbox at the beginning of the dialogue. When ELVIS was tested with the learned strategies, task completion increased from .85 (training) to .94 (test) ($p < .07$), and user satisfaction increased from 27.5 (training) to 31.7 (test) ($p < .05$).

2.2. NJFun Experiments

NJFun is a spoken dialogue system that provides information about leisure activities in New Jersey. The goal of the dialogue was to acquire values for three variables: type of activity, location, and time of day. A sample dialogue is in Figure 3. NJFun had more limited capabilities than ELVIS in order to keep the state space small. It provided no help messages and it could not undo or repeat previous actions. In addition, if the system failed to acquire the information for one of the variables, the dialogue would continue with the variable unbound and items matching any variable value would be returned by the database.

NJFun was trained via interactions varying system, user and mixed-initiative strategies and explicit confirmation vs. no confirmation. In training, 54 users each performed 6 information access tasks with the system. For testing, 21 users performed the same 6 tasks. The reward was binary task completion, an all or nothing measure calculated automatically from the system logs. This reward was used because task completion for the training version of NJFun was only .52. The results were that binary task completion improved to .64 (test), ($p < .06$).

The system learned that the optimal use of initiative is to begin with user initiative, then back off to either mixed or system initiative when re-asking for an attribute. The optimal confirmation strategy is to mainly confirm at lower confidence values, but the point where confirmation becomes unnecessary differs across attributes, and depends on other features of the state besides confidence (e.g., grammar and history). Both the initiative and confirmation results suggest that the beginning of the dialogue was the most problematic for NJFun. The dialogue in Figure 3 uses the optimal strategies.

3. Applications of Boosting to Spoken Language Generation

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

Figure 4. Alternative sentence plan realizations for a combination of confirming and requesting information. H = human rating, RB = RankBoost score.

We conducted two sets of experiments using boosting. The SPoT trainable generator targeted information gathering in the travel domain for the AT&T Communicator dialogue system (Rambow et al., 2001). Examples are in Figure 4. The SPaRKy generator (Stent et al., 2004) focused on utterances to RECOMMEND a restaurant or COMPARE sets of restaurants for the MATCH system. Example recommendations are in Figure 1. To apply boosting to spoken language generation requires:

1. A generator that can produce multiple alternative responses at any point in the dialogue;
2. Ratings by human users of each alternative set of responses in a dialogue context. The associated ratings induces a partial order over the set of possible responses;

3. Alternative responses are represented by a set of features describing various aspects of the response, or the context for the response;

4. Given the ratings and the features for each response, the training method learns how to reproduce the partial order (ranking) of responses.

The feedback is used to *evaluate* whether the stochastic generator produces sets of high quality alternatives, and to *train* the generator to produce utterances matching this criterion. Given this setup, system developers can choose whether the feedback of multiple users is averaged and used for training (Rambow et al., 2001; Stent et al., 2004), or whether the system is trained for an individual user (Mairesse and Walker, 2005).

The ratings can represent any metric associated with the possible response, e.g. coherence, information quality, social appropriateness. We used a metric called *informational coherence*, collected via user feedback. Users are shown response variants and told: For each variant, please rate to what extent you agree with the statement *The utterance is easy to understand, well-formed and appropriate to the dialogue context.*

If the generator produced alternatives that varied along a different evaluative scale, feedback relevant to optimizing that scale could be elicited. For example, if the generator produced utterances that varied from formal to friendly, users could state their degree of agreement with the statement: *The system is very friendly.* Alternatively, the feedback (objective function) need not be elicited as a subjective judgement. As with reinforcement learning, if the goal were to generate utterances that can be processed with low cognitive effort, an instrumented measure of cognitive effort, such as reaction time, could be used as the objective function.

The training method utilizes the RankBoost algorithm (Freund et al., 1998), where 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 values of feature counts, for a large set of features generated automatically from the linguistic representations that the generator uses. For example, one indicator function is

$$h_{100}(x) = \begin{cases} 1 & \text{if LEAF-ASSERT-RECO-BEST}(x) \geq 1 \\ 0 & \text{otherwise} \end{cases}$$

So $h_{100}(x) = 1$ if the leftmost leaf of the linguistic tree representation is the assertion that *Chanpen Thai has the best overall quality among the selected restaurants*, as in Alts 6 and 1 in Figure 1. A single parameter α_s is associated with each indicator function, 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 alternatives with the goal of duplicating the ranking found in the training data, and the training examples are used to set the parameter values α_s. The user ratings are converted into a training set $\mathcal{T}$ of *ordered pairs* of examples x, y:

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

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 (cases where ranking scores disagree with human judgments) 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. The value of the loss function is reported as the experimental testing error rate.

3.1. Results

Using cross-validation, we conducted experiments for both SPoT and SPaRKy as described more fully elsewhere (Rambow et al., 2001; Stent et al., 2004), testing whether the Rankboost algorithm can reproduce the human rankings. Figures 1 and 4 show the generated alternatives, the human rankings (on a scale of 1 to 5) and the ranking values (on a scale of 0 to 1) from the trained ranker. It is easy to see from inspection that the training method is fairly accurate at reproducing the human rankings. Quantitative results are in Table 3.1. The error rates for SPoT and SPaRKy are for a generator trained with the mean feedback of two judges, whereas SPaRKy Individual represents results for training individualized generators. To put these results in context, the average ranking loss for a set of 50 randomly generated rankings of 20 alternatives is also shown. It can be seen that the training method is a large improvement over a random baseline.

Table I. Error rates for different generators

	Random	SPoT	SPaRKy	SPaRKy Individual
Ranking Error	.50	0.19	0.23	.17

4. Discussion and Future Work

This paper argues that language generation algorithms cannot be evaluated or trained by direct comparison with corpora. We presented two alternative methods for training dialogue managers and spoken language generators, reinforcement learning and boosting, and summarized experimental results using these methods. While particular scalar metrics were used in the presented experiments, the strength of these approaches is that any evaluation metric can be used for the objective function these training methods require. The reinforcement learning experiments illustrate how different evaluation metrics and thus different objective functions are appropriate at different stages of system development.

In terms of useful resources for applying such techniques, both methods require an explicit representation of alternative choices that a system has at a particular point in a dialogue. In the reinforcement learning experiments, these choices were built into the dialogue manager before any training data was collected, but given a method for calculating dialogue state, a corpus could be augmented with such representations (Scheffler and Young, 2002). In the boosting experiments, we collected dialogue interactions using a modified version of the template-based generator of AT&T communicator that logged a conceptual representation of each system utterance, then used this to generate alternatives that were then rated in context by users to form the basis for our experiments. A similar technique was used in the MATCH system. Thus a valuable resource for this type of technique are corpora annotated with semantic representations to support the generation of alternative realizations, and annotations indicating human ratings of these alternatives.

References

André, E., T. Rist, S. van Mulken, M. Klesen, and S. Baldes: 2000, 'The automated design of believable dialogues for animated presentation teams'. *Embodied conversational agents* pp. 220–255.

Barzilay, R., N. Elhadad, and K. R. McKeown: 2002, 'Infering Strategies for Sentence Ordering in Multidocument Summarization'. *Journal of Artificial Intelligence Research* **17**, 35–55.

Freund, Y., R. Iyer, R. E. Schapire, and Y. Singer: 1998, 'An efficient boosting algorithm for combining preferences'. In: *Machine Learning: Proceedings of the Fifteenth International Conference.*

Hirschman, L.: 2000, 'Evaluating Spoken Language Interaction: Experiences from the DARPA Spoken Language Program 1990–1995'. In: S. Luperfoy (ed.): *Spoken Language Discourse.* Cambridge, Mass.: MIT Press.

Lapata, M.: 2003, 'Probabilistic text structuring: Experiments with sentence ordering'. In: *Proceedings of the ACL.*

Litman, D. J., M. S. Kearns, S. Singh, and M. A. Walker: 2000, 'Automatic Optimization of Dialogue Management'. In: *Proc. COLING 2000.*

Mairesse, F. and M. Walker: 2005, 'Training an Individualized Generator for Spoken Dialogue'. In: *In submission.*

Rambow, O., M. Rogati, and M. Walker: 2001, 'Evaluating a Trainable Sentence Planner for a Spoken Dialogue Travel System'. In: *Proceedings of the Meeting of the Association for Computational Lingustics, ACL 2001.*

Reiter, E.: 2002, 'Should Corpora be Gold Standards for NLG?'. In: *Proceedings of the 11th International Workshop on Natural Language Generation.* pp. 97–104.

Scheffler, K. and S. Young: 2002, 'Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning'. In: *Human Language Technology Conference.*

Stent, A., R. Prasad, and M. Walker: 2004, 'Trainable Sentence Planning for Complex Information Presentation in Spoken Dialogue Systems'. In: *Meeting of the Association for Computational Linguistics.*

Sutton, R. S. and A. G. Barto: 1998, *Reinforcement Learning.* MIT Press.

Walker, M. A.: 2000, 'An Application of Reinforcement Learning to Dialogue Strategy Selection in a Spoken Dialogue System for Email'. *Journal of Artificial Intelligence Research* **12**, 387–416.

Walker, M. A., D. Litman, C. A. Kamm, and A. Abella: 1997, 'PARADISE: A General Framework for Evaluating Spoken Dialogue Agents'. In: *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics, ACL/EACL 97.* pp. 271–280.

Yeh, C.-L. and C. Mellish: 1997, 'An Empirical Study on the Generation of Anaphora in Chinese'. *Computational Linguistics* **23-1**, 169–190.