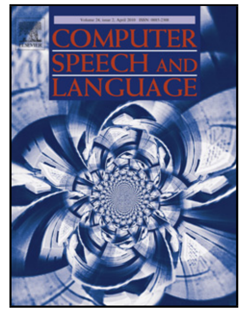
Learning what to say and how to say it: joint optimization of spoken dialogue management and Natural Language Generation

Oliver Lemon

Accepted Manuscript



Title: Learning what to say and how to say it: joint optimization of spoken dialogue management and Natural Language Generation

Author: Oliver Lemon

PII: S0885-2308(10)00036-7
DOI: doi:10.1016/j.csl.2010.04.005
Reference: YCSLA 452

To appear in:

Received date: 6-5-2009
Revised date: 7-12-2009
Accepted date: 1-4-2010

Please cite this article as: Lemon, O., Learning what to say and how to say it: joint optimization of spoken dialogue management and Natural Language Generation, *Computer Speech & Language* (2008), doi:10.1016/j.csl.2010.04.005

Learning what to say and how to say it: joint optimisation of spoken dialogue management and Natural Language Generation

Oliver Lemon

Heriot Watt University
Edinburgh

Abstract

This paper argues that the problems of dialogue management (DM) and Natural Language Generation (NLG) in dialogue systems are closely related and can be fruitfully treated statistically, in a joint optimisation framework such as that provided by Reinforcement Learning (RL). We first review recent results and methods in automatic learning of dialogue management strategies for spoken and multimodal dialogue systems, and then show how these techniques can also be used for the related problem of Natural Language Generation. This approach promises a number of theoretical and practical benefits such as fine-grained adaptation, generalisation, and automatic (global) optimisation, and we compare it to related work in statistical/trainable NLG. A demonstration of the proposed approach is then developed, showing combined DM and NLG policy learning for adaptive information presentation decisions. A joint DM and NLG policy learned in the framework shows a statistically significant 27% relative increase in reward over a baseline policy, which is learned in the same way only without the joint optimisation. We thereby show that that NLG problems can be approached statistically, in combination with dialogue management decisions, and we show how to jointly optimise NLG and DM using Reinforcement Learning.

Key words: Dialogue Systems, Natural Language Generation, Reinforcement Learning

Email address: o.lemon@hw.ac.uk (Oliver Lemon)
URL: http://sites.google.com/site/olemon/ (Oliver Lemon)

Preprint submitted to Computer Speech and Language *December 7, 2009*

1. Introduction

Much recent work has focussed on the problems of automatically learning Dialogue Management strategies or "dialogue policies" for spoken and multimodal dialogue systems. A dialogue policy defines what the system should do or say next at any particular point in a conversation, at the level of Dialogue Acts. For example, in a particular context, whether the dialogue manager should decide to "inform(price=cheap)"[1] or "present_results(item3, item4, item7)"[2].

Natural Language Generation (NLG) in dialogue is often characterised as choosing "how" to say something once "what to say" has been determined. For example, depending on the domain of the system, "inform(price=cheap)" might be uttered as "This restaurant is economically priced" or "That's a cheap place to eat", and "present_results(item3,item4,item7)" could be uttered as, "There are 3 results for your search. The first is El Bar on Quixote street, the second is Pancho Villa on Elm Row, and the third is Hot Tamale on Quality Street" or else perhaps by comparing the 3 restaurants in some manner.

In principle, NLG in dialogue thus comprises a wide variety of decisions, ranging over content structuring, choice of referring expressions, use of ellipsis, aggregation, and choice of syntactic structure, to the choice of intonation markers for synthesised speech. In computational dialogue systems, "what to say" is usually determined by a dialogue manager (DM) component, via planning, hand-coded rules, finite state machines, or learned policies, and "how to say it" is then very often defined by simple templates or hand-coded rules which define appropriate word strings to be sent to a speech synthesizer or screen.

This paper argues that the statistical optimisation approaches that have recently proven successful for Dialogue Management decisions can also be used for the problems of Natural Language Generation. Moreover, I argue that DM and NLG decisions should not be made in isolation, and show that a combined, jointly optimised policy, is better than one which optimises the decisions separately.

Previous statistical approaches to NLG are reviewed in section 3, but

[1]For example by saying "This restaurant is good for people on a tight budget."

[2]For example, saying "Kebab Mahal, The Red Fort, and The Mosque Kitchen are all well-rated indian restaurants in the centre of Edinburgh"

none of them have explored NLG as statistical *planning*. Aspects of NLG have been treated as planning problems before [14, 35], but not statistically, and prior dialogue-related work in statistical planning (e.g. Reinforcement Learning) has dealt only with policies for planning dialogue acts in information gathering, and has not been applied to NLG decisions themselves (though see [28] for initial work in multimodal generation).

In principle, learning approaches to NLG could have several key advantages over template-based and rule-based approaches (we discuss "trainable" NLG in section 3.3) in dialogue systems:

- the ability to adapt to fine-grained changes in dialogue context,
- a data-driven development cycle,
- provably optimal action policies with a precise mathematical model for action selection, and
- the ability to generalise to unseen dialogue states.

We first discuss recent methods and results which illustrate these advantages for the problem of dialogue management policies (section 2.1), and extend them to NLG in the demonstration system described in section 4.

2. Background: learning approaches to Dialogue Management

The basic model for the approaches we will discuss below is the Markov Decision Process or MDP. Here a stochastic system interacting with its environment (in our case, the user of the dialogue system) through its actions is described by a number of states $\{s_i\}$ in which a given number of actions $\{a_j\}$ can be performed. In a dialogue system, the states represent the possible dialogue contexts (e.g. how much information we have so far obtained from the user, what has previously been said in the conversation [6, 9][3] etc.), and the actions are now system dialogue actions.

Each state-action pair is associated with a transition probability $\mathcal{T}_{ss'}$: the probability of moving from state s at time t to state s' at time $t+1$ after

[3]Note that a common misunderstanding is that the Markov Property constrains models of dialogue state to exclude the dialogue history. However, we can employ variables in the current state which represent features of the dialogue history.

having performed action a when in state s. These probabilities depend on how users respond to the system's actions. The transitions are also associated with a reinforcement signal (or "reward") r_{t+1} describing how good the result of action a was when performed in state s. In dialogue these reward signals are most often associated with task completion and dialogue length, and in most work they have been defined by the system designer (e.g. 100 points for successful task completion, -1 per turn) to optimise some a priori objective of the dialogue system. However, as we shall discuss, recent work has developed data-driven methods for defining reward functions [41, 28].

To control a system described in this way, one then needs a strategy or policy π mapping all states to actions: $\pi(s) = P(a|s)$. In this framework, a Reinforcement Learning agent is a system aiming at optimally mapping states to actions, i.e. finding the best strategy π^* so as to maximize an overall reward R which is a function (most often a weighted sum) of all the immediate rewards. In dialogue (and many other problems) the reward for an action is often not immediate, but is *delayed* until successful completion of a task. Of course, actions affect not only the immediate reward, but also the next state and thereby all subsequent rewards.

In general, then, we are trying to find an action policy π which maximises the value $Q^\pi(s, a)$ of choosing action a in state s, which is given by the Bellman equation:

$$Q^\pi(s,a) = \sum_{s'} \mathcal{T}_{ss'}[\mathcal{R}_{ss'} + \gamma V^\pi(s')] \tag{1}$$

(Here we denote the expected immediate reward by $\mathcal{R}_{ss'}$, γ is a discount factor between 0 and 1, $V^\pi(s)$ is the value of state s' according to π, see [36]).

If the transition probabilities are known, an analytic solution can be computed by dynamic programming. Otherwise the system has to learn the optimal strategy by a trial-and-error process, for example using Reinforcement Learning methods [36] as we do in this paper. Trial-and-error search and delayed rewards are the two main features of Reinforcement Learning.

With these concepts in place, we can discuss recent advances made in the application of such models to real dialogue management problems.

2.1. Recent advances in Learning Dialogue Policies

Following the initial proposal and developments of the Reinforcement Learning approach to dialogue management from 1997-2002 [18, 40, 19, 42, 32, 33], a number of research groups have further developed the approach in several directions, which have tackled the following central problems:

- learning with large state spaces,
- learning from small initial data sets,
- learning the reward function, and
- combining hand-coded/rule-based and learned behaviours.

An extensive data-driven methodology has now been developed [25, 28, 31] to address these issues, using the following techniques:

- function approximation to reduce the size of the state space,
- learning realistic user simulations from small data sets,
- regression techniques for data-driven discovery of reward functions,
- Hierarchical Reinforcement Learning, allowing some decisions to be hand-programmed, while others are learned.

We now briefly describe each of these advances.

2.1.1. Function approximation and generalisation methods

Function approximation methods address the need to generalise from small amounts of data to large state spaces. [9] demonstrated the effectiveness of this method using the COMMUNICATOR corpus. Here, a large feature-based representation of dialogue context was employed, which in principle generated over 10^{386} possible dialogue states. There is therefore a very high chance that a state encountered in testing will not be exactly the same as any state encountered in training data. Linear function approximation was used to map from a vector of real valued features $f(s)$ for each state s to a vector of estimates $Q(s, a)$, with one estimate for each a. The trained parameters of the linear function are a vector of weights for each action a. This approximation method has the effect of treating two states as similar if they share features, and then the estimates learned for one state also affect all similar states. This has the effect that a policy learned in this manner generalises to previously unseen states.

2.1.2. User simulations from small data sets

Given the large number of possible policies requiring exploration for RL, it is often said that training with real users would be "cruel and unusual punishment" for humans. For this reason, and for reasons of speed and cost, simulated users are commonly employed to train DM policies for tens of thousands of simulated dialogues. However, the transition probabilities encoded by such simulations must be realistic if the learned policy is to be good for real dialogues. Again, there is a problem of learning an appropriate simulation from small amounts of real data, typically gathered in "Wizard-of-Oz" data collections. [28] shows how to learn multimodal dialogue strategies by interaction with a simulated environment which is "bootstrapped" from small amounts of Wizard-of-Oz (WOZ) data. They use a cluster-based user simulation method to deal with the small initial data set, see [26].

2.1.3. Discovering Reward functions

A common criticism levelled at RL approaches is that the system simply learns what it is told to by the reward function. This is true in the sense that the reward function specifies precisely what the system should achieve, but it does not specify *how* that should be achieved. Precisely how to achieve the best reward can be a matter of balancing many competing trade-offs, and may lead to non-trivial policies. A deeper criticism is that if the reward function is hand-coded, or specified by intuition, then that is little different to specifying intuitive hand-coded rules for solving the DM problem [23]. Indeed, "programming by reward" can be similar to hand-coding rules, but recent work has shown how to specify reward functions in a data-driven manner. The original idea is due to Walker [41], who uses a regression analysis to discover a (weighted) linear combination of objective features (e.g. dialogue length, task completion) that is predictive of subjective user ratings (e.g. "Future Use" of the system). This method thereby reveals how measurable features of the dialogue context can be weighted and combined to define reward signals that are discovered in the data, rather than specified by hand. It has been used successfully in [27, 28, 30, 31].

2.1.4. Hierarchical MDPs

It is often argued that there is little point in learning obvious decisions (e.g. to greet at the start of a conversation) and that learned policies should be re-usable in different systems. Using hierarchical MDPs addresses both of these issues [17, 3]. It allows a mixture of hand-coded and learned decisions, and the structuring of decision problems into a hierarchy, where different state features may be available at each level.

The learning system presented in section 4 illustrates a simple hierarchical MDP model. The top level of the hierarchy makes DM decisions, while the lower levels handle NLG. Hierarchical MDPs allow joint optimisation through decomposition of complex decision problems, and thereby dramatically reduce the size of the state space needed for learning.

3. Prior work in Natural Language Generation

There are 3 main approaches to generating system utterances in dialogue systems: template-based NLG, conventional NLG as developed in the text generation literature [24], and more recently, trainable generation [1, 5, 34, 37, 38].

3.1. Templates

Template-based generation is typically used in industrial dialogue systems, and even in most state-of-the-art research systems. However, this approach requires that new templates be created by hand for each application, and severely limits that system's ability to adapt to dialogue context or user preferences, due to the practical constraints of having to write different templates for each possible combination of feature values.

3.2. Conventional NLG

Conventional NLG typically follows a pipeline architecture consisting of three main modules [24]:

- (i) a text planner which performs content selection and discourse structuring,
- (ii) a sentence planner which selects attributes for referring expressions, aggregates content into sentence-size units, and selects lexical items, and
- (iii) a surface realiser which converts sentence plans into natural language.

This approach has been successfully applied in systems that tailor their presentations to the user's preferences [2, 4, 20, 39] and to the dialogue context [10], but these systems generally use rules that are specifically hand-crafted for a particular domain. A major problem with these standard NLG approaches is that hand-coded rules, manually-set thresholds, and templates all severely limit the adaptivity that can be achieved in NLG, both in the amount of adaptivity possible and the ability to adapt to fine-grained changes in the dialogue context, user behaviour, or environment (e.g. noise levels). The standard approaches are limited by the expertise of the system designer, and the adaptivity that they can encode in their rules or templates. Statistical approaches in general promise a more practical, effective, and theoretically well-founded approach to adaptivity in NLG, because they are not limited by human design capacities, and can be trained from data. However, in practice, the content planning and sentence planning components usually consist of domain-specific rules, or general rules that are tuned for the specific domain. Conventional NLG approaches can also be too slow for real-time dialogue applications [34]. There has therefore been recent interest in statistical methods in the area of "trainable" NLG.

In addition, several recent studies have shown that when information is tailored to users' preferences and previous dialogue behaviour, users judge systems' contributions as significantly easier to understand and as "better" information [4, 21, 39].

3.3. Trainable NLG

Here, automatic techniques are used to train NLG modules, or to adapt them to specific domains and/or types of user. However, this work has focussed on local optimisation through supervised learning, and has not explored global decision-theoretic planning approaches such as Reinforcement Learning.

Early work here focused on supervised learning of how to produce surface forms from sentence plans, using overgeneration and ranking, using either bigram language models [22], or ranking rules learned from a corpus of manually ranked training examples [37]. More recent work has extended this approach to sentence-planning [34].

In [34], given a content plan (the propositions to express and discourse relations among them), a generator first produces a set of text-plan trees, consisting of speech acts to be communicated, and the rhetorical relations between them. For each of these, a set of candidate sentence plans are generated by a heuristically ordered set of clause-combining operations. The sentence plan ranker is then trained by using the RankBoost algorithm to learn a set of ranking rules from a manually labelled set of examples.

For content selection, recent research has shown that given a corpus of texts and the database of facts or events it describes, content selection rules can be learned [1, 5]. In this work, content selection has been treated as a binary classification task. Here, semantic units in the database are first aligned with sentences in the corpus, and then classification is used to learn whether or not a semantic unit should be included in the text.

However, as explained above, these types of supervised learning used for NLG do not model the required optimisation and planning of sequences of actions-in-context which we propose to capture with RL techniques. An interesting issue for future work is how these types of classifier-based learning can be integrated with the MDP approach demonstrated here.

4. The joint optimisation model: combined DM and NLG as statistical planning

This paper now treats NLG as a statistical planning and optimisation problem using decision theory in the framework of Markov Decision Processes (MDPs), similar to [28]. The main advance here is to treat aspects of NLG within the same MDP-based planning and learning frameworks as have been successfully applied in speech recognition and dialogue management, for example [18, 40, 33, 42].

In prior work on dialogue strategy learning, only dialogue acts (e.g. greet, ask slot, explicit confirm) chosen by the system have been optimised. Here we go beneath the level of dialogue acts to plan NLG actions such as content structuring. Several key questions thus arise – how to represent different NLG actions for planning, what context features and state representations are important in the MDP, and what reward signals can be used to optimise NLG? We now present a fully worked example to show the model in use: learning for adaptive Information Presentation.

One of the classic problems in NLG is how to present one or more items to a user, for example by simply listing them, contrasting them in respect of some attributes, or clustering similar items together (e.g. [20, 38]). For example the system may present items like so:

- LIST: "There are four hotels meeting your criteria. The first is the Royal, the second is ..."

- CONTRAST: "The Oak is an expensive central hotel. The Royal is cheap but is not central. ..."
- CLUSTER: "There are 7 expensive hotels and 11 cheap ones, ...".

We will model these decisions in a hierarchical MDP, and solve it using trial-and-error exploration, using Reinforcement Learning methods. First, we model the states of the system.

4.1. State space

In this example we will have 3 search constraint slots that the user can fill (for instance `food type, location, price range` for a restaurant search application, or `artist, album, genre` for music browsing). These slots can be either filled or confirmed. Confirmed slots have 100% chance of being correct, and filled slots only 80% chance, thereby modelling noise in the speech recognition environment (see also [16, 28]). In addition we will model the number of "hits" or search results returned by the system after every user turn – this will be a number from 0 to 100. Importantly, the baseline system does not have access to this feature when deciding how to present items.

4.2. Action set

See figure 1 for the hierarchical structure of the actions available to the system, representing the combined dialogue management (DM) and NLG task as an inter-related planning problem.

Here we see a top-level "skill" (ConductDialogue) responsible for dialogue management choices in the MDP, and a second level skill (PresentInfo) which is responsible for the NLG choices. ConductDialogue governs the standard dialogue management options, and decomposes into the 4 possible action choices (or "Means") AskASlot, ImplicitConfirm_and_AskASlot, ExplicitConfirm, and PresentInfo. This allows the system to choose between these 4 types of dialogue act at any time. More interestingly, for the NLG component of the system we have implemented possible 3 action choices (or "Means") for information presentation under PresentInfo: ContrastItems, ListItems, and ClusterItems. ListItems is just the standard list content structuring operator, while ContrastItems and ClusterItems are actions which structure the items presented to the users by contrasting them and clustering them respectively, as shown above.

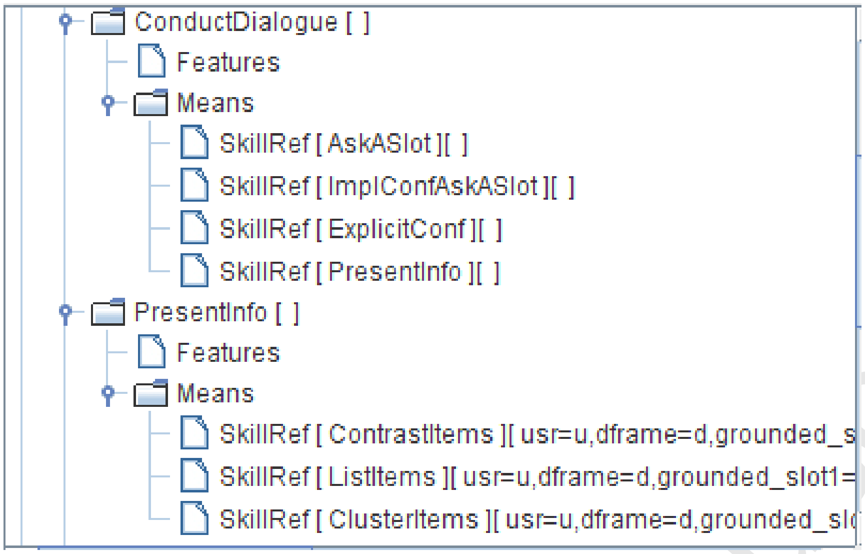


Figure 1: A Hierarchical Plan for NLG and DM joint optimisation

4.3. Reward function

Now that we have our states and actions, we need to define a Reward signal (or "Objective function") for the learning system. This directs the learner in terms of its overall goals (e.g. short dialogues where users rate information presentation highly), while it is up to the learner to find an action policy which meets these goals. What makes the learning problem interesting is that these goals contain conflicting "trade-offs" that the system must learn to balance, based on the state that it is in. For example, the goal to have short dialogues conflicts with the goal to get reliable (i.e. confirmed) search constraints from the user and to present small numbers of items. The learning problem here is then for the system to decide at each turn whether to ask for more information/constraints, confirm (explicitly or implicitly) the existing information/constraints, or to List, Contrast, or Cluster the current items returned from the DB. Note that the system can decide to immediately present information in some way to the user even if not all slots are filled or confirmed. This leaves open the option for the system to exploit a "good" information presentation situation (such as having only 2 database items to tell the user about via a Contrast) even if the DM situation (e.g. having only 2 filled slots) is not in itself very rewarding. In this way the NLG and DM decisions are jointly optimised in this setup.

For training a system to be deployed with real users, this reward/objective function would be developed based on a PARADISE-style [41] analysis of a small amount of Wizard-of-Oz data [40, 27]. Here, to prove the concept of statistical planning for NLG, we simply show that the learner can jointly optimise the NLG and dialogue management decisions based on a complex reward signal. Nothing depends on the particular values chosen here – they are for illustration only and can be estimated from suitable data.

The overall reward for each dialogue conducted by the system has 3 components: *completion reward, turn penalty*, and *presentation reward/penalty*. Turn penalty is simply -1 per system turn. The completion reward is the % probability that the items presented to the user correctly meet their actual search constraints, and is therefore a function of the number of filled or confirmed slots. For example, if all 3 slots are confirmed, then (in this noise model) we have 100% chance of having the search constraints correct. The number of filled/confirmed slots stochastically determines the number of items that the system can present to the user if it decides to enter the presentation phase. For example if 3 slots are filled, then 0-10 items will be presented to the user, if 2 slots are filled 0-20 items are retrieved, if 1 slot, 0-100 items. Again, in a real application, these distributions would be estimated from data.

The presentation reward (PR) for each information presentation action is defined as follows, for i = number of items to be presented:

- ListItems: $0 \leq i \leq 3 : PR = 100; 4 \leq i \leq 8 : PR = 0; 9 \leq i : PR = -100$
- ContrastItems: $i \leq 1 : PR = -100; 2 \leq i \leq 6 : PR = 300; 7 \leq i : PR = -100$
- ClusterItems: $0 \leq i \leq 5 : PR = -100; 5 \leq i \leq 8 : PR = 0; 9 \leq i : PR = 300$

This range of rewards/penalties, together with those for filled and confirmed slots and dialogue length provides a complex environment within which the learner must explore different trade-offs.

4.4. Environment and User simulation

For training a policy given this definition of states, actions, and rewards, we also need an environment simulation that responds appropriately to system actions. Here the environment is not only the user, but also the database from which items are retrieved for presentation to the user. For policy exploration we use a simple bigram stochastic user simulation with probabilities estimated from COMMUNICATOR data, similar to [8]. At each system turn, a number of database hits is randomly determined as a function of the number of filled search constraint slots, as described above. Note that this user simulation does not need to respond directly to the NLG decisions of the system, since the dialogue closes as soon as the system decides to present information (in whatever manner) to the user. A central open question for this type of MDP model of NLG is how to develop "good" user simulations that are sensitive to system NLG choices, see [11, 13].

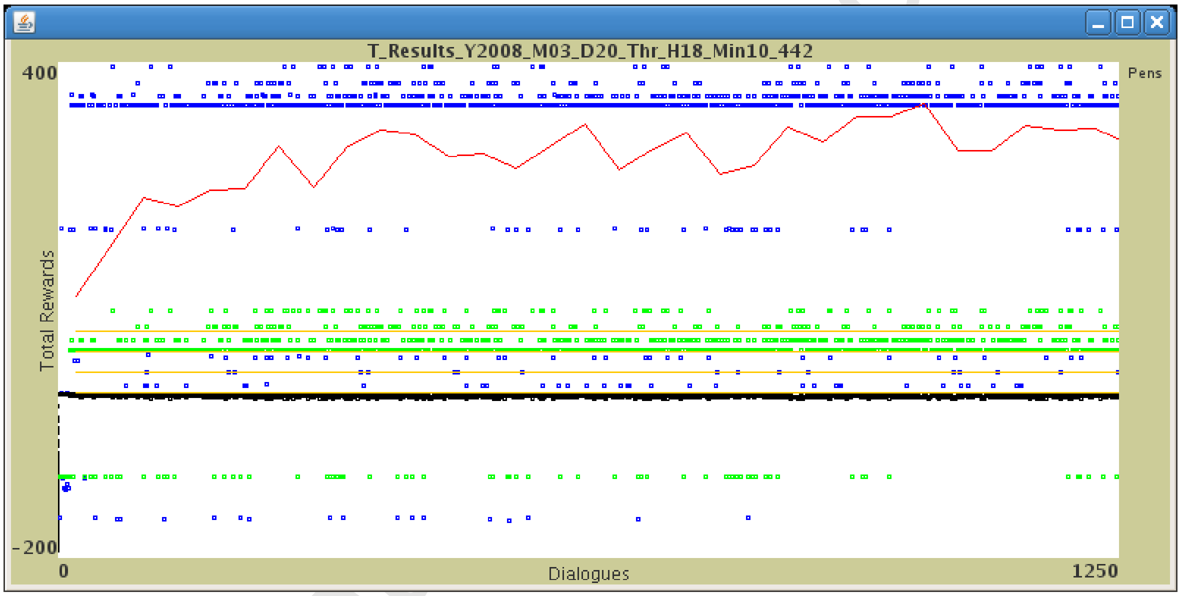


Figure 2: Training the adaptive NLG policy (red lines show average reward over windows of 50 dialogues, blue dots show total reward per dialogue, black dots show length penalty, and green dots show task completion.

4.5. The Baseline policy

In contrast to other work on policy learning, which typically uses hand-coded systems for comparison, we choose a more challenging baseline. This is because hand-coded policies have been shown to be inferior to learned policies in numerous studies, e.g. [18, 33, 16, 40], and also, because our task here is a combination of dialogue management and NLG, we do not want the NLG results to be contaminated by an inferior hand-coded dialogue management policy. We therefore choose to compare against a baseline policy learned for the same problem domain, (i.e. the policy is learned using the same setup and parameters), but where the learner uses the *average* most rewarding action for the NLG component (in this case, Cluster items). We can think of this as the RL analogue of a majority class baseline. This baseline policy does not have access to the "DB hits" feature for decisions under PresentInfo (it does have this feature for the top level decisions though), so it learns the average best NLG action rather than attempting to learn the best NLG action for each possible number of DB hits. It therefore is unable to jointly optimise DM and NLG decisions.

4.6. Training the policies

We use a hierarchical SARSA Reinforcement Learning algorithm [36] with linear function approximation to train the policies. Figure 2 shows learning for the joint DM&NLG problem[4].

Here we see that after 1250 training dialogues the system has learned to find a high average reward for the combined NLG and DM problem. At the start of training the system explores bad actions in some states, for example the minimum reward gained in early training is -153, obtained by contrasting more than 7 items (-100) when only 1 slot is filled (-50) after 3 system turns (-3). However, by the end of this training run, the system is able to consistently obtain the best possible rewards given the dialogue situation, for example gaining a top reward of 396 for either Contrast or Cluster of appropriate numbers of items (+300), when all slots are confirmed (+100) in system 4 turns (-4). Where no +300 presentation reward is possible (i.e. $i = 1, 7$, or 8) the system has learned to Cluster or List (when i=1) the items after filling and confirming all slots. A similar graph can be shown for training the Baseline policy.

4.7. Testing

We trained both policies multiple times until convergence (approx. 10K cycles), selected the best policy in each case, and tested them (with stochastic simulated users) for 550 test dialogues each. Figure 3 shows the performance

[4]In the training/testing graphs red lines show average reward over windows of 50 dialogues, and for each dialogue blue dots show total reward (including NLG reward), black dots show length penalty, and green show completion reward per dialogue.

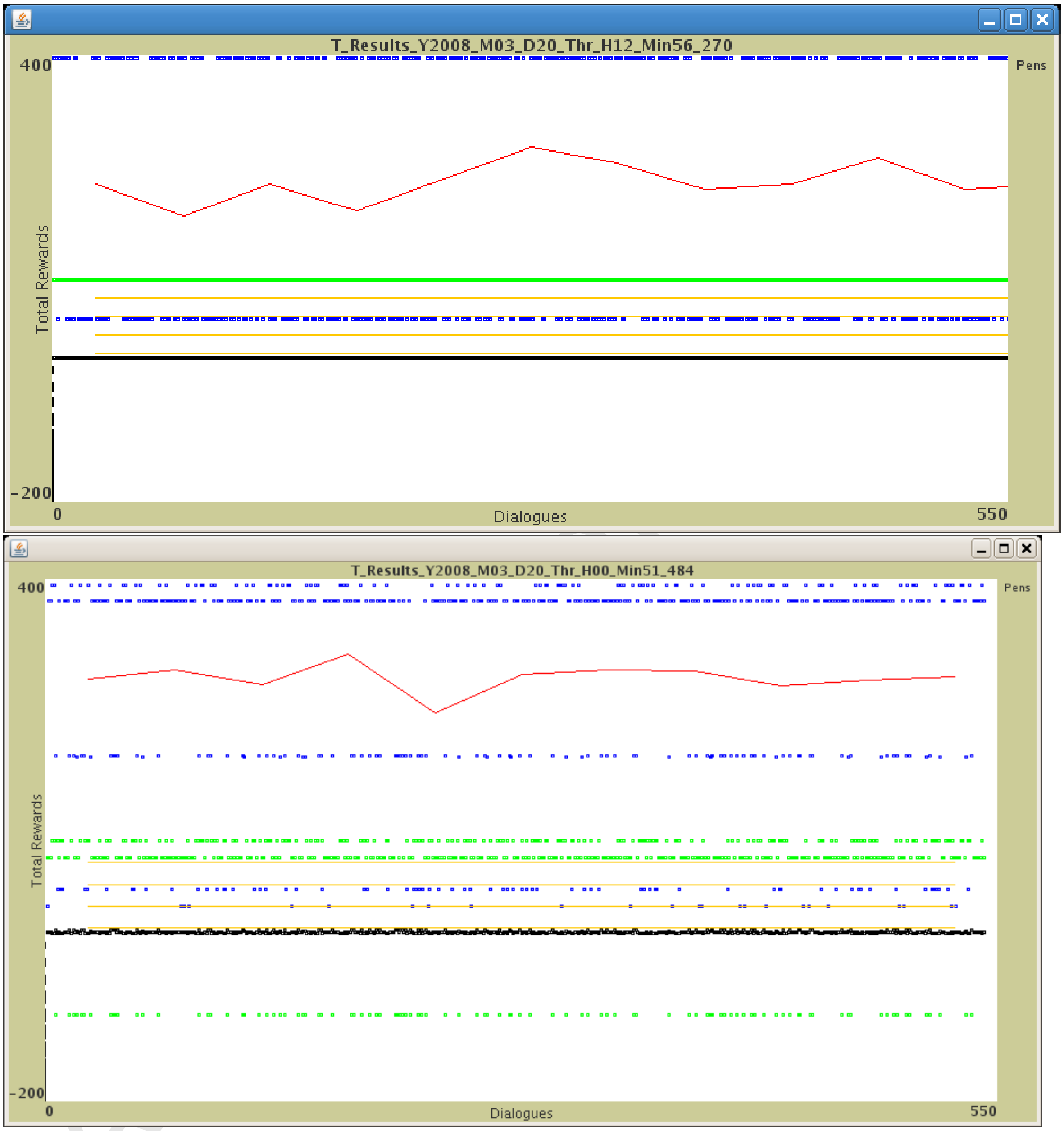


Figure 3: Testing the Baseline (top, av. =224.5) and joint (bottom, av. =286.9) DM&NLG policies

Policy	Av. Reward	Av. length
Baseline Learned	224.5	4.0
Joint DM & NLG Learned	286.9*	4.98

Table 1: Results: learned baseline vs. joint DM&NLG policies. ($* = p < 0.001$)

of the 2 policies during testing (top= baseline DM&NLG , bottom = joint DM&NLG), and the results are presented in table 1.

These results demonstrate a relative increase in reward of 27.8% for the jointly optimised system. The DM&NLG system has learned fine-grained local trade-offs for its NLG decisions, which are not available to the baseline system. For alternative rule-based baselines, please see [29].

So what has been learned? Here is an example dialogue with the DM&NLG system:

`System`: How can I help you? `(greet)`

`User:` I want a cheap chinese restaurant. (2 slots filled, 2 database items returned)

`System:` Ok. The Golden Wok is cheap and central, and the Noodle bar is cheap but in the south of the city `(Contrast)`

Here we can see that the DM&NLG policy can decide to present information when it is particularly advantageous, even when the information gathering stage of the system is not complete. The Baseline learns a similar policy, but is not sensitive to the number of DB hits when choosing *how* to present the information. Note that we could of course train only the NLG part ofthe system, using a fixed DM policy, as in [11].

5. Summary and Future Directions

This paper demonstrates a new data-driven method where the DM and NLG components of dialogue systems can be automatically and jointly trained and globally optimised before deployment. We show that a combined, jointly optimised policy, is better than one which optimises the decisions separately.

We first surveyed recent advances in learning approaches to DM, and standard approaches to NLG, and described general advantages offered by statistical planning models together with solution methods such as Reinforcement Learning. We gave a brief description of MDP models. In section 4 we cast a standard NLG problem as a Hierarchical MDP, defining the state space, action set, and reward function. We saw how Reinforcement Learning can be used to solve this NLG problem at the same time as optimising dialogue management. We then evaluated the jointly learned DM&NLG policy versus a learned baseline policy lacking the joint optimisation. The results showed a significant relative increase in reward of 27.8% for the DM&NLG system. When given a reward signal that provides feedback on content structuring choices (List, Contrast, Cluster) the system learns to avoid bad decisions (e.g. listing lots of items, clustering small numbers of items, contrasting too few or too many items) and to choose the best NLG option available depending on the number of database items returned by the system at any time.

This demonstrates that the proposed approach brings a number of theoretical and practical benefits such as fine-grained adaptation, and automatic optimisation. Note that the use of function approximation also allows these policies to generalise to states/contexts that have not been encountered during training. Function approximation methods also allow scaling up to very large state-action spaces, see [9].

5.1. *Future work, open questions*

It would be interesting to approach other NLG decisions in this way. By using MDPs to represent other NLG problems we can move to a situation where determination of the best lexical items and referring expressions [11] to use in a system utterance, as well as the best syntactic structure and intonation pattern, are all determined by learned strategies, developed by reward-driven learning based on real data [29, 12]. Reinforcement Learning could also be applied to decisions of when and how to use anaphora and ellipsis. Future challenges also include modelling the hierarchical structure of NLG problems using additional hierarchical MDPs, and modelling complex effects of NLG choices on dialogue context using larger feature sets. In addition, we can explore different action sets, such as sequences of Information Presentations (e.g. Summary + Recommend [29]). An interesting open question is to what extent such methods can be scaled up to large action sets of both DM and NLG actions, and the correspondingly more complex state spaces required to make such decisions. Again, function approximation techniques have been shown to scale well in related work [9, 31].

Overall, this leads us to investigate a new development process for NLG components of dialogue systems, whereby the more adaptive NLG components of new dialogue systems can be automatically trained and optimised before deployment, and can then be allowed to adapt online to user feedback (through continued monitoring of rewards). Moreover, due to the use of state generalisation techniques such as function approximation, reasonable NLG decisions will be possible in previously unseen and unplanned-for situations.

An open question for this type of model is how to develop "good" user simulations that are sensitive to system NLG choices [13]. Another important topic is how the classifier-based learning techniques of "trainable" NLG [1, 5, 34, 38] can be integrated with the MDP approach proposed here. Other avenues to explore are how interactive alignment [7] and semantic coordination in dialogue [15] can be modelled in this framework.

Acknowledgements

The research leading to these results has received funding from the EPSRC (project nos. EP/E019501/1 and EP/G069840/1) and from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 216594 (CLASSiC project www.classic-project.org)

References

[1] Regina Barzilay and Mirella Lapata. Collective content selection for concept-to-text generation. In *Proceedings of EMNLP*, 2005.

[2] Giuseppe Carenini and Johanna D. Moore. Generating and evaluating evaluative arguments. *Artificial Intelligence*, 170(11):925–952, 2006.

[3] Heriberto Cuayáhuitl. *Hierarchical Reinforcement Learning for Spoken Dialogue Systems*. PhD thesis, Edinburgh University, School of Informatics, 2009.

[4] Vera Demberg and Johanna D. Moore. Information presentation in spoken dialogue systems. In *Proceedings of EACL*, 2006.

[5] Pablo A. Duboue and Kathleen R. McKeown. Statistical acquisition of content selection rules for natural language generation. In *Proceedings of EMNLP*, 2003.

[6] Matthew Frampton and Oliver Lemon. Learning more effective dialogue strategies using limited dialogue move features. In *Proceedings of ACL*, pages 185–192, Sydney, 2006.

[7] Simon Garrod and Martin Pickering. Toward a mechanistic psychology of dialogue: The interactive alignment model. In *Proceedings of BI-dialog*, 2001.

[8] Kallirroi Georgila, James Henderson, and Oliver Lemon. User simulation for spoken dialogue systems: Learning and evaluation. In *Proceedings of Interspeech/ICSLP*, pages 1065–1068, 2006.

[9] James Henderson, Oliver Lemon, and Kallirroi Georgila. Hybrid reinforcement/supervised learning of dialogue policies from fixed datasets. *Computational Linguistics*, 34(4):487–512, 2008.

[10] Amy Isard, Jon Oberlander, Ion Androutsopoulos, and Colin Matheson. Speaking the users' languages. *IEEE Intelligent Systems Magazine*, 18(1):40–45, 2003.

[11] Srini Janarthanam and Oliver Lemon. A User Simulation Model for learning Lexical Alignment Policies in Spoken Dialogue Systems. In *European Workshop on Natural Language Generation*, 2009.

[12] Srini Janarthanam and Oliver Lemon. A Wizard-of-Oz environment to study Referring Expression Generation in a Situated Spoken Dialogue Task. In *European Workshop on Natural Language Generation*, 2009.

[13] Srinivasan Janarthanam and Oliver Lemon. User simulations for online adaptation and knowledge-alignment in Troubleshooting dialogue systems. In *Proceedings of SEMdial*, pages 149–156, 2008.

[14] Alexander Koller and Matthew Stone. Sentence generation as planning. In *Proceedings of ACL*, 2007.

[15] Staffan Larsson. Coordinating on ad-hoc semantic systems in dialogue. In *Proceedings of DECALOG*, 2007.

[16] Oliver Lemon and Xingkun Liu. Dialogue policy learning for combinations of noise and user simulation: transfer results. In *SIGdial*, 2007.

[17] Oliver Lemon, Xingkun Liu, Daniel Shapiro, and Carl Tollander. Hierarchical Reinforcement Learning of Dialogue Policies in a development environment for dialogue systems: REALL-DUDE. In *Proceedings of Brandial, the 10th SemDial Workshop on the Semantics and Pragmatics of Dialogue, (demonstration systems)*, 2006.

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

[19] E. Levin, R. Pieraccini, and W. Eckert. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing*, 8(1):11–23, 2000.

[20] Johanna Moore, Mary Ellen Foster, Oliver Lemon, and Michael White. Generating tailored, comparative descriptions in spoken dialogue. In *Proc. FLAIRS*, 2004.

[21] Johanna Moore, Mary Ellen Foster, Oliver Lemon, and Michael White. Generating tailored, comparative descriptions in spoken dialogue. In *The 17th International FLAIRS Conference (Florida Artifical Intelligence Research Society)*, 2004.

[22] Alice Oh and Alexander Rudnicky. Stochastic natural language generation for spoken dialog systems. *Computer, Speech & Language*, 16(3/4):387–407, 2002.

[23] Tim Paek and David Maxwell Chickering. The Markov Assumption in spoken dialogue management. In *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*, 2005.

[24] Ehud Reiter and Robert Dale. *Building Natural Language Generation Systems*. CUP, 2000.

[25] Verena Rieser. *Bootstrapping Reinforcement Learning-based Dialogue Strategies from Wizard-of-Oz data*. PhD thesis, Saarbruecken dissertations in Computational Linguistics and Language Technology (Vol 28), 2008.

[26] Verena Rieser and Oliver Lemon. Cluster-based User Simulations for Learning Dialogue Strategies and the SUPER evaluation metric. In *Proceedings of Interspeech/ICSLP*, pages 1766–1769, 2006.

[27] Verena Rieser and Oliver Lemon. Automatic Learning and Evaluation of User-Centered Objective Functions for Dialogue System Optimisation. In *Proceedings of LREC*, 2008.

[28] Verena Rieser and Oliver Lemon. Learning Effective Multimodal Dialogue Strategies from Wizard-of-Oz data: Bootstrapping and Evaluation. In *Proceedings of ACL*, 2008.

[29] Verena Rieser and Oliver Lemon. Natural language generation as planning under uncertainty for spoken dialogue systems. In *EACL*, 2009.

[30] Verena Rieser and Oliver Lemon. Learning human multimodal clarification strategies for use in dialogue systems. *J. Natural Language Engineering*, page (to appear), 2010.

[31] Verena Rieser and Oliver Lemon. Learning and evaluation of dialogue strategies for new applications: a fully data-driven methodology. *Computational Linguistics*, (under review).

[32] Satinder Singh, Michael Kearns, Diane Litman, and Marilyn Walker. Reinforcement learning for spoken dialogue systems. In *Advances in Neural Information Processing Systems*, 2000.

[33] Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artificial Intelligence Research (JAIR)*, 2002.

[34] Amanda Stent, Rashmi Prasad, and Marilyn Walker. Trainable sentence planning for complex information presentation in spoken dialog systems. In *Association for Computational Linguistics*, 2004.

[35] Matthew Stone, Christine Doran, Bonnie Webber, Tonia Bleam, and Martha Palmer. Microplanning with communicative intentions: the SPUD system. *Computational Intelligence*, 19(4):311–381, 2003.

[36] Richard Sutton and Andrew Barto. *Reinforcement Learning*. MIT Press, 1998.

[37] M. Walker, O. Rambow, and M. Rogati. Spot: A trainable sentence planner. In *In Proc. of the NAACL,*, 2001.

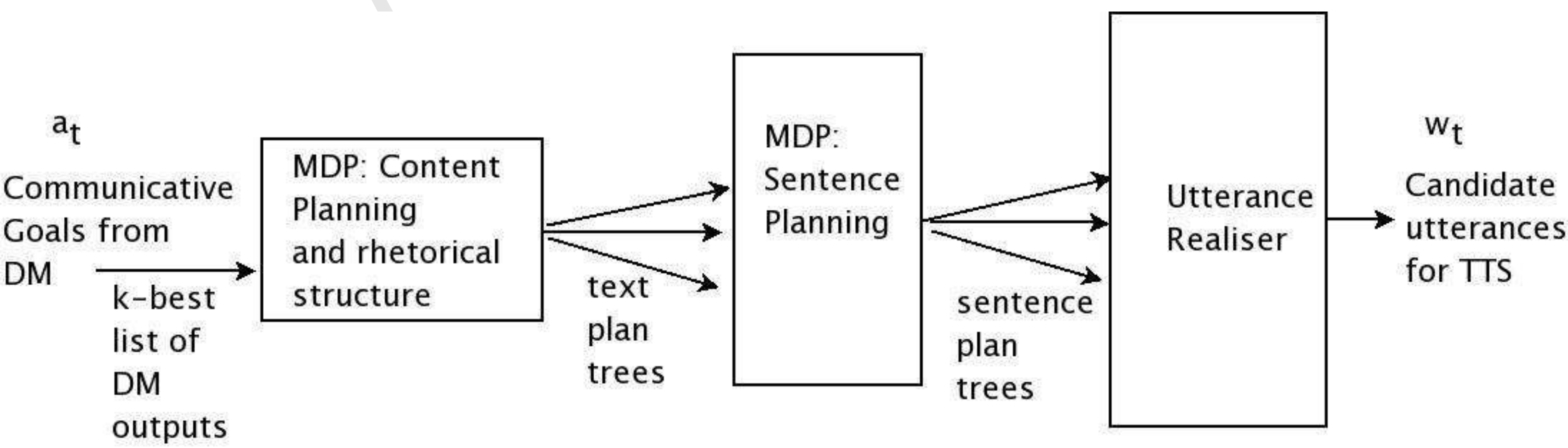
[38] Marilyn Walker, Amanda Stent, François Mairesse, and Rashmi Prasad. Individual and domain adaptation in sentence planning for dialogue. *Journal of Artificial Intelligence Research (JAIR)*, 30:413–456, 2007.

[39] Marilyn Walker, S. Whittaker, A. Stent, P. Maloor, J. Moore, M. Johnston, and G. Vasireddy. User tailored generation in the match multimodal dialogue system. *Cognitive Science*, 28:811–840, 2004.

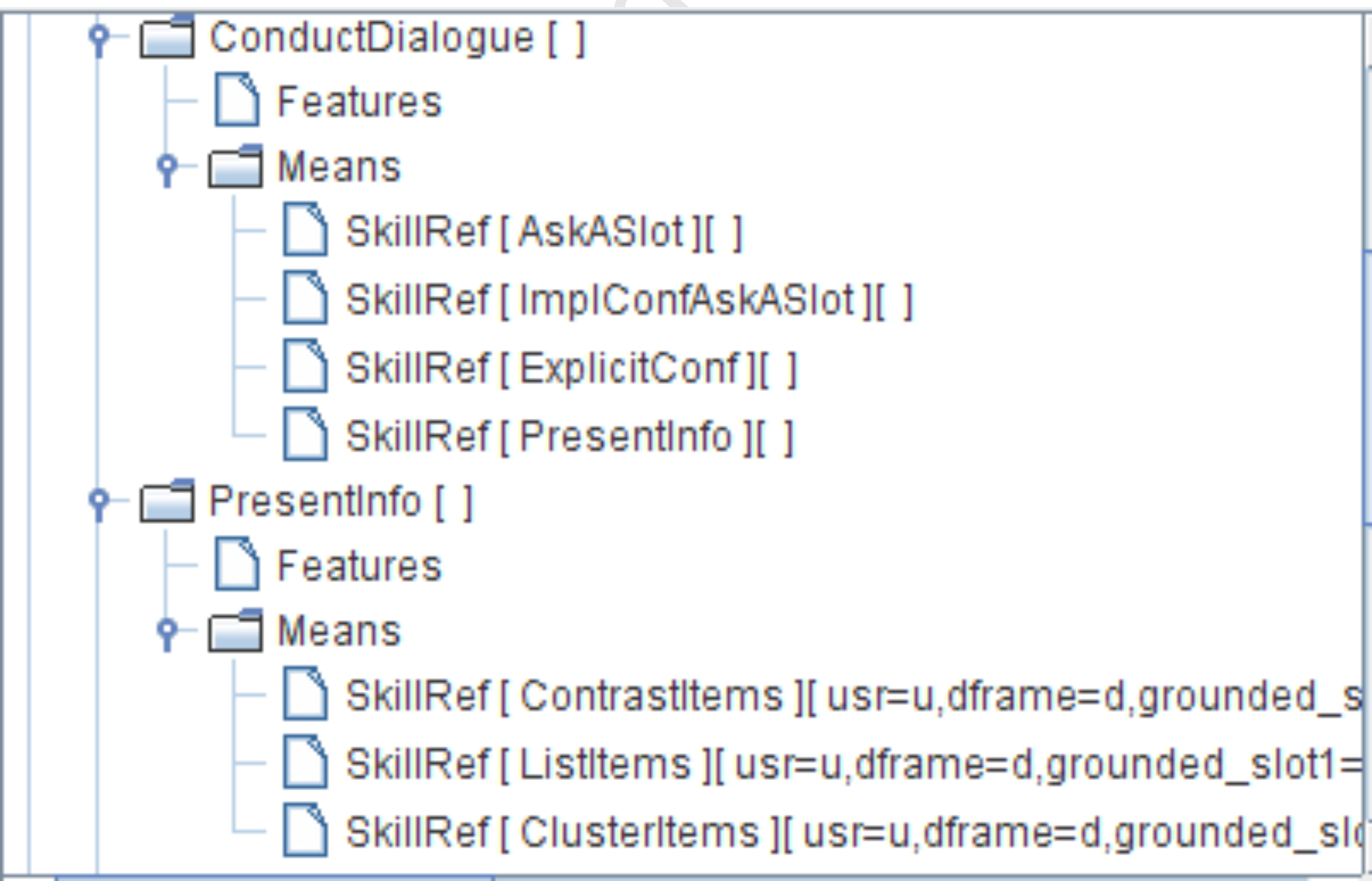
[40] Marilyn A. Walker, Jeanne C. Fromer, and Shrikanth Narayanan. Learning optimal dialogue strategies: a case study of a spoken dialogue agent for email. In *Proceedings of ACL*, 1998.

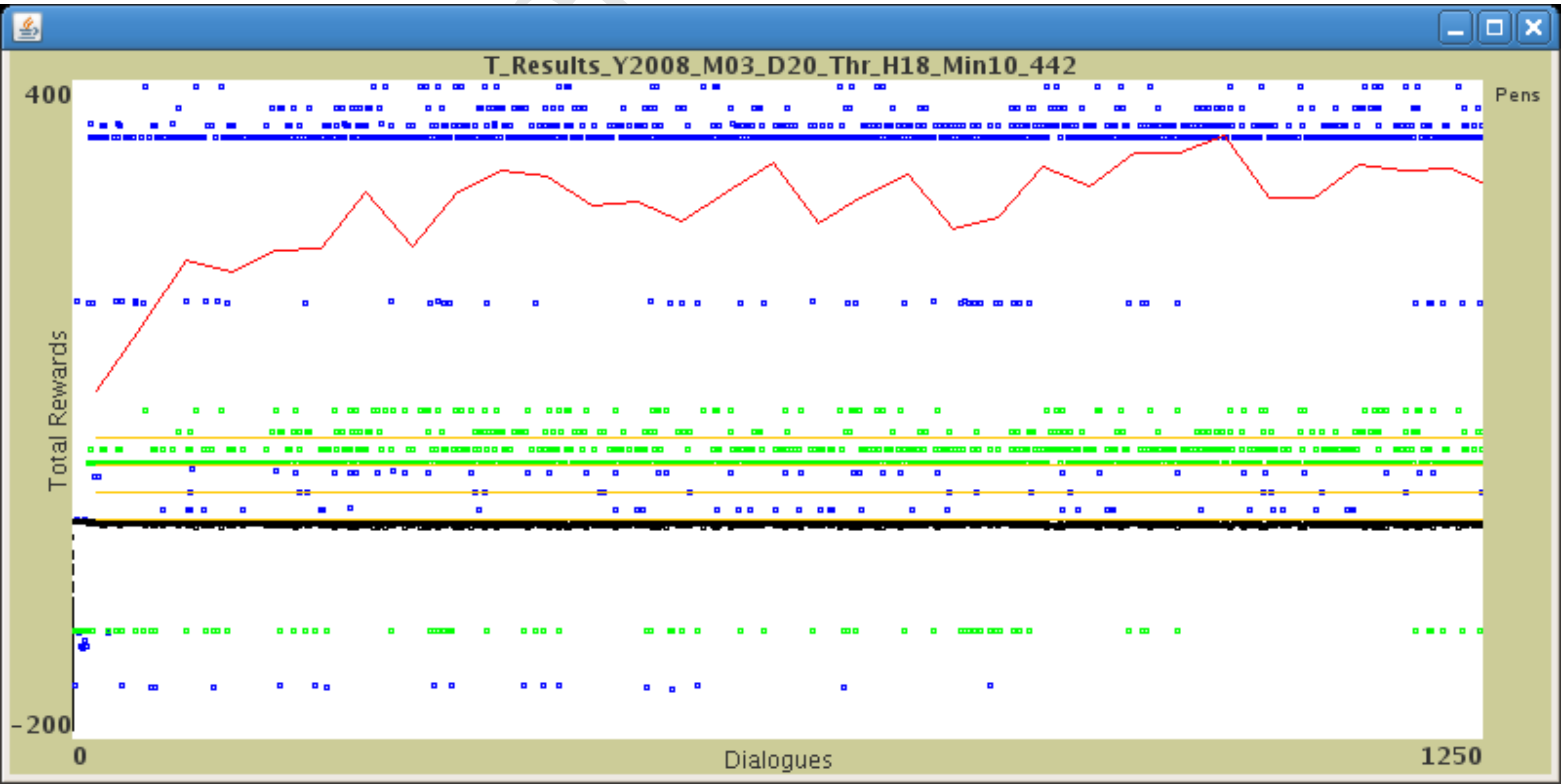
[41] Marilyn A. Walker, Candace A. Kamm, and Diane J. Litman. Towards Developing General Models of Usability with PARADISE. *Natural Language Engineering*, 6(3), 2000.

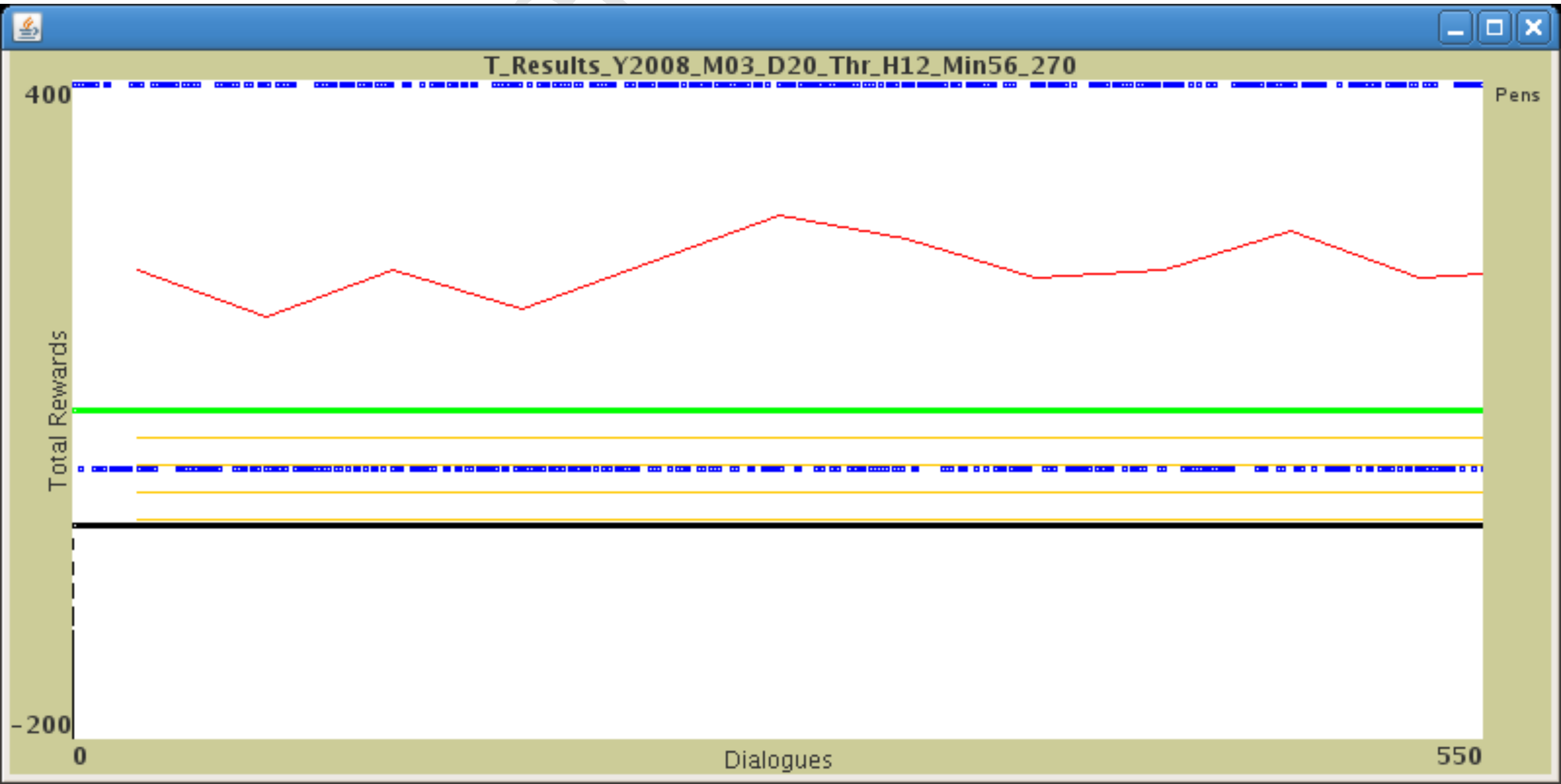
[42] Steve Young. Probabilistic methods in spoken dialogue systems. *Philosophical Transactions of the Royal Society (Series A)*, 358(1769):1389–1402, 2000.



a_t
Communicative Goals from DM
k-best list of DM outputs
MDP: Content Planning and rhetorical structure
text plan trees
MDP: Sentence Planning
sentence plan trees
Utterance Realiser
w_t
Candidate utterances for TTS


ConductDialogue [ ]
Features
Means
SkillRef [AskASlot][ ]
SkillRef [ImplConfAskASlot][ ]
SkillRef [ExplicitConf][ ]
SkillRef [PresentInfo][ ]
PresentInfo [ ]
Features
Means
SkillRef [ContrastItems][usr=u,dframe=d,grounded_s
SkillRef [ListItems][usr=u,dframe=d,grounded_slot1=
SkillRef [ClusterItems][usr=u,dframe=d,grounded_sl


T_Results_Y2008_M03_D20_Thr_H18_Min10_442
400
-200
Total Rewards
Pens
0
Dialogues
1250


T_Results_Y2008_M03_D20_Thr_H12_Min56_270
400
-200
Total Rewards
0
Dialogues
550
Pens