Optimising Natural Language Generation Decision Making For Situated Dialogue

Nina Dethlefs
Department of Linguistics,
University of Bremen
dethlefs@uni-bremen.de

Heriberto Cuayáhuitl
German Research Centre
for Artificial Intelligence (DFKI)
heriberto.cuayahuitl@dfki.de

Jette Viethen
Centre for Language Technology
acquarie University
jviethen@ics.mq.edu.au

Abstract

Natural language generators are faced with a multitude of different decisions during their generation process. We address the joint optimisation of navigation strategies and referring expressions in a situated setting with respect to task success and human-likeness. To this end, we present a novel, comprehensive framework that combines supervised learning, Hierarchical Reinforcement Learning and a hierarchical Information State. A human evaluation shows that our learnt instructions are rated similar to human instructions, and significantly better than the supervised learning baseline.

1 Introduction

Natural Language Generation (NLG) systems are typically faced with a multitude of decisions during their generation process due to nondeterminacy between a semantic input to a generator and its realised output. This is especially true in situated settings, where sudden changes of context can occur at anytime. Sources of uncertainty include (a) the situational context, such as visible objects, or task complexity, (b) the user, including their behaviour and reactions, and (c) the dialogue history, including shared knowledge or patterns of linguistic consistency (Halliday and Hasan, 1976) and alignment (Pickering and Garrod, 2004).

Previous work on context-sensitive generation in situated domains includes Stoia et al. (2006) and Garoufi and Koller (2010). Stoia et al. present a supervised learning approach for situated referring expression generation (REG). Garoufi and Koller use techniques from AI planning for the combined generation of navigation instructions and referring expressions (RE). More generally, the NLG problem of non-deterministic decision making has been addressed from many different angles, including PENMAN-style choosers (Mann and Matthiessen, 1983), corpus-based statistical knowledge (Langkilde and Knight, 1998), tree-based stochastic models (Bangalore and Rambow, 2000), maximum entropy-based ranking (Ratnaparkhi, 2000), combinatorial pattern discovery (Duboue and McKeown, 2001), instance-based ranking (Varges, 2003), chart generation (White, 2004), planning (Koller and Stone, 2007), or probabilistic generation spaces (Belz, 2008) to name just a few.

More recently, there have been several approaches towards using Reinforcement Learning (RL) (Rieser et al., 2010; Janarthanam and Lemon, 2010) or Hierarchical Reinforcement Learning (HRL) (Dethlefs and Cuayáhuitl, 2010) for NLG decision making. All of these approaches have demonstrated that HRL/RL offers a powerful mechanism for learning generation policies in the absence of complete knowledge about the environment or the user. It overcomes the need for large amounts of hand-crafted knowledge or data in rule-based or supervised learning accounts. On the other hand, RL can have difficulties to find an optimal policy in a large search space, and is therefore often limited to small-scale applications. Pruning the search space of a learning agent by including prior knowledge is therefore attractive, since it finds solutions faster, reduces computational demands, incorporates expert knowledge, and scales to complex problems. Suggestions to use such prior knowledge include Litman et al. (2000) and Singh et al. (2002), who hand-craft rules of prior knowledge obvious to the system designer. Cuayáhuitl (2009) suggests using Hierarchical Abstract Machines to partially pre-specify dialogue strategies, and Heeman (2007) uses a combination of RL and Information State (IS) to also pre-specify dialogue strategies. Williams (2008) presents an approach of combining Partially-Observable Markov Decision Processes with conventional dialogue systems. The Information State approach is well-established in dialogue management (e.g., Bohlin et al. (1999) and Larsson and Traum (2000)). It allows the system designer to specify dialogue strategies in a principled and systematic way. A disadvantage is that random design decisions need to be made in cases where the best action, or sequence of actions, is not obvious.

The contribution of this paper consists in a comprehensive account of constrained Hierarchical Reinforcement Learning through a combination with a hierarchical Information State (HIS), which is informed by prior knowledge induced from decision trees. We apply our framework to the generation of navigation strategies and referring expressions in a situated setting, jointly optimised for task success and linguistic consistency. An evaluation shows that humans prefer our learnt instructions to the supervised learning-based instructions, and rate them equal to human instructions. Simulation-based results show that our semi-learnt approach learns more quickly than the fully-learnt baseline, which makes it suitable for large and complex problems. Our approach differs from Heeman's in that we transfer it to NLG and to a hierarchical setting. Although Heeman was able to show that his combined approach learns more quickly than pure RL, it is limited to small-scale systems. Our 'divide-and-conquer' approach, on the other hand, scales up to large search spaces and allows us to address complex problems.

2 The Generation Tasks

2.1 The GIVE-2 Domain

Our domain is the generation of navigation instructions and referring expressions in a virtual 3D world in the GIVE scenario (Koller et al., 2010). In this task, two people engage in a 'treasure hunt', where an instruction giver (IG) navigates an instruction follower (IF) through the world, pressing a sequence of buttons and completing the task by obtaining a trophy. Pairs take part in three dialogues (in three different worlds); after the first dialogue, they switch roles. The GIVE-2 corpus (Gargett et al., 2010) provides transcripts of such dialogues in English and German. For this paper, we complemented the English dialogues of the corpus with a set of semantic annotations.[1] The feature set is organised in five groups (Table 1). The first two groups cover manipulation instructions (i.e., instructions to press a button), including distractors[2] and landmarks (Gargett et al., 2010). The third group describes high- and low-level navigation, the fourth group describes the user. The fifth group finally contains grammatical information.

2.2 Navigation and Manipulation Instructions

Navigation instructions can take many forms, even for the same route. For example, a way to another room can be described as 'go to the room with the lamp', 'go left and through the door', or 'turn 90 degrees, left, straight'. Choosing among these variants is a highly context- and speaker-dependent task. Figure 1 shows the six user strategies we identified from the corpus based on an analysis of the combination of navigation level (*'high'* vs. *'low'*) and content (*'destination'*, *'direction'*, *'orientation'*, *'path'*, *'straight'*). User models are based on the navigation level and content decisions made in a sequence of instructions, so that different sequences, with a certain distribution, lead to different user model classifications. The proportions are shown in Figure 1. We found that 75% of all speakers use the same strategy in consecutive rounds/games. 62.5% of pairs are consistent over all three dialogues, indicating inter-speaker alignment. These high measures of human consistency suggest that this phenomenon is worth modelling in a learning agent, and therefore provides the motivation of including linguistic consistency in our agent's behaviour. Manipulation instructions were treated as an REG task, which needs to be sensitive to the properties of the referent and distractors (e.g, size, colour, or spatial relation

[1]The annotations are available on request.

[2]Distractors are objects of the same type as the referent.

ID	Feature	Type	Description
f_1	absolute_property(referent)	$boolean$	Is the colour of the referent mentioned?
f_2	absolute_property(distractor)	$boolean$	Is the colour of the distractor mentioned?
f_3	discriminative_colour(referent)	$boolean$	Is the colour of the referent discriminating?
f_4	discriminative_colour(distractor)	$boolean$	Is the colour of the distractor discriminating?
f_5	mention(distractor)	$boolean$	Is a distractor mentioned?
f_6	first_mention(referent)	$boolean$	Is this the first reference to the referent?
f_7	mention(macro_landmark)	$boolean$	Is a macro (non-movable) landmark mentioned?
f_8	mention(micro_landmark)	$boolean$	Is a micro (movable) landmark mentioned?
f_9	num(distractors)	$integer$	How many distractors are present?
f_{10}	num(micro_landmarks)	$integer$	How many micro landmarks are present?
f_{11}	spatial_rel(referent,obj)	$string$	Which spatial relation(s) are used in the RE?
f_{12}	taxonomic_property(referent)	$boolean$	Is the type of the distractor mentioned?
f_{13}	within_field_of_vision(referent)	$boolean$	Is the referent within the user's field of vision?
f_{14}	mention(colour, lm)	$boolean$	Is the colour of a macro- / micro lm mentioned?
f_{15}	mention(size, lm)	$boolean$	Is the size of a macro- / micro lm mentioned?
f_{16}	abstractness(nav_instruction)	$string$	Is the instruction *explicit* or *implicit*?
f_{17}	content(nav_instruction)	$string$	Vals: *destination, direction, orientation, path, straight*
f_{18}	level(nav_instruction)	$string$	Is the instruction *high-* or *low-level*?
f_{19}	position(user)	$string$	Is the user *on_track* or *off_track*?
f_{20}	reaction(user)	$string$	Vals: *take_action, take_wrong_action, wait, req_help*
f_{21}	type(user)	$string$	Vals: *likes_waiting, likes_exploring, in_between*
f_{22}	waits(user)	$boolean$	Is the user waiting for the next instruction?
f_{23}	model(user)	$string$	User model/navig. strategy used (cf. Fig.1)?
f_{24}	actor(instruction)	$boolean$	Is the actor of the instruction inserted?
f_{25}	mood(instruction)	$boolean$	Is the mood of the instruction inserted?
f_{26}	process(instruction)	$boolean$	Is the process of the instruction inserted?
f_{27}	locational_phrase(instruction)	$boolean$	Is the loc. phrase (path, straight, etc.) inserted?

Table 1: *Corpus annotation features that were used as knowledge of the learning agent and the Information State. Features are presented in groups, describing the properties of referents in the environment (f_1...f_{13}) and their distractors (f_{14}...f_{15}), features of high- and low-level navigation (f_{16}...f_{18}), the user (f_{19}...f_{23}), and grammatical information about constituents (f_{24}...f_{27}).*

with respect to the referent) to be natural and distinguishing. We also considered the visual salience of objects, and the type of spatial relation involved, since recent studies indicate the potential relevance of these features (Viethen and Dale, 2008). Given these observations, we aim to optimise the **task success** and **linguistic consistency** of instructions. Task success is measured from user reactions after each instruction (Section 5.1). Linguistic consistency is achieved by rewarding the agent for generating instructions that belong to the same user model as the previous one. The agent has the same probability for choosing any pattern, but is then rewarded for consistency. Table 3 (in Section 5.2) presents an example dialogue generated by our system.

3 Constrained Hierarchical Reinforcement Learning for NLG

3.1 Hierarchical Reinforcement Learning

Our idea of *language generation as an optimisation problem* is as follows: given a set of generation states, a set of actions, and an objective reward function, an optimal generation strategy maximises the objective function by choosing the actions leading to the highest reward for every reached state. Such states describe the system's knowledge about

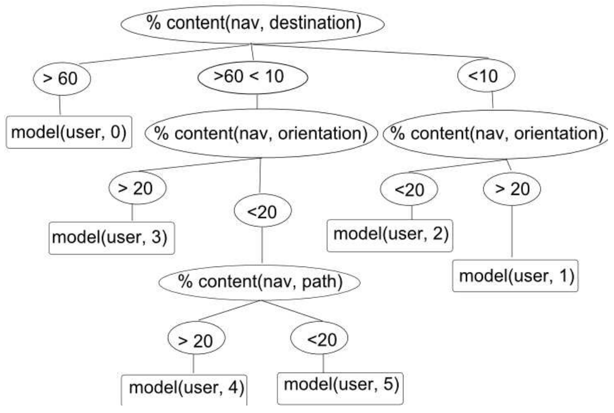


Figure 1: *Decision tree for the classification of user models (UM) defined by the use of navigation level and content. UM 0=high-level, UM 1=low-level (LL), UM 2=orientation-based LL, UM 3=orientation-based mixture (M), UM 4=path-based M, UM 5=pure M.*

the generation task (e.g. navigation strategy, or referring expressions). The action set describes the system's capabilities (e.g. *'use high level navigation strategy'*, *'mention colour of referent'*, etc.). The reward function assigns a numeric value for each action taken. In this way, language generation can be seen as a finite sequence of states, actions and rewards $\{s_0, a_0, r_1, s_1, a_1, ..., r_{t-1}, s_t\}$, where the goal is to find an optimal strategy automatically. To do this we use RL with a divide-and-conquer approach in order to optimise a hierarchy of generation policies rather than a single policy. The hierarchy of RL agents consists of L levels and N models per level, denoted as M^i_j, where $j \in \{0, ..., N-1\}$ and $i \in \{0, ..., L-1\}$. Each agent of the hierarchy is defined as a Semi-Markov Decision Process (SMDP) consisting of a 4-tuple $< S^i_j, A^i_j, T^i_j, R^i_j >$. S^i_j is a set of states, A^i_j is a set of actions, T^i_j is a transition function that determines the next state s' from the current state s and the performed action a, and R^i_j is a reward function that specifies the reward that an agent receives for taking an action a in state s lasting τ time steps. The random variable τ represents the number of time steps the agent takes to complete a subtask. Actions can be either primitive or composite. The former yield single rewards, the latter correspond to SMDPs and yield cumulative discounted rewards. The goal of each SMDP is to find an optimal policy that maximises the reward for each visited state, according to $\pi^{*i}_j(s) = \arg\max_{a \in A^i_j} Q^{*i}_j(s, a)$, where $Q^{*i}_j(s, a)$ specifies the expected cumulative reward for executing action a in state s and then following policy π^{*i}_j. We use HSMQ-Learning (Dietterich, 1999) for learning a hierarchy of generation policies. This hierarchical approach has been applied successfully to dialogue strategy learning by Cuayáhuitl et al. (2010).

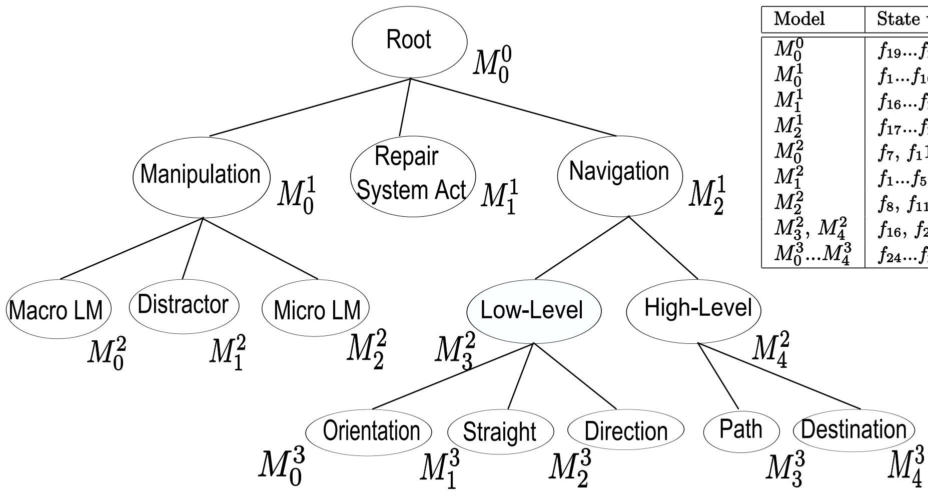
3.2 Information State

The notion of an Information State has traditionally been applied to dialogue, where it encodes all information relevant to the current state of the dialogue. This includes, for example, the context of the interaction, participants and their beliefs, and the status of grounding. An IS consists of a set of *informational components*, encoding the information of the dialogue, *formal representations* of these components, a set of *dialogue moves* leading to the update of the IS, a set of *update rules* which govern the update, and finally an *update strategy*, which specifies which update rule to apply in case more than one applies (Larsson and Traum (2000), p. 2-3). In this paper, we apply the theory of IS to language generation. For this purpose we define the informational components of an IS to represent the (situational and linguistic) knowledge of the generator (Section 4.2). Update rules are triggered by generator actions, such as the decision to insert a new constituent into the current logical form, or the decision to prefer one word order sequence over another. We use the DIPPER toolkit (Bos et al., 2003)[3] for our implementation of the IS.

3.3 Combining Hierarchical Reinforcement Learning and Information State

Previous work has suggested the HSMQ-Learning algorithm for optimizing text generation strategies (Dethlefs and Cuayáhuitl, 2010). Because such an algorithm uses all available actions in each state, an important extension is to constrain the actions available with some prior expert knowledge, aiming to combine behaviour specified by human designers and behaviour automatically inferred by reinforcement learning agents. To that end, we sug-

[3]http://www.ltg.ed.ac.uk/dipper



Model	State variables (features)
M_0^0	$f_{19}...f_{22}$
M_0^1	$f_1...f_{10}$, $f_{12}...f_{13}$, $f_{19}...f_{20}$, f_{22}
M_1^1	$f_{16}...f_{22}$
M_2^1	$f_{17}...f_{23}$
M_0^2	f_7, $f_1 1$, $f_{13}...f_{15}$
M_1^2	$f_1...f_5$, f_{11}, f_{13}
M_2^2	f_8, f_{11}, f_{13}, f_{20} ,f_{22}
M_3^2, M_4^2	f_{16}, f_{21}, f_{20} ,$f_{22}...f_{23}$
$M_0^3...M_4^3$	$f_{24}...f_{27}$

Figure 2: (Left:) Hierarchy of learning agents executed from top to bottom for generating instructions. (Right:) State representations for the agents shown in the hierarchy on the left. The features $f_1...f_{27}$ refer back to the features used in the annotation given in the first column of Table 1. Note that agents can share information across levels.

gest combining the Information State approach with hierarchical reinforcement learning. We therefore re-define the characterisation of each Semi-Markov Decision Process (SMDP) in the hierarchy as a 5-tuple model $M_j^i =< S_j^i, A_j^i, T_j^i, R_j^i, I_j^i >$, where S_j^i, A_j^i, T_j^i and R_j^i are as before, and the additional element I_j^i is an Information State used as knowledge base and rule-based decision maker. In this extended model, action selection is based on a constrained set of actions provided by the IS update rules. We assume that the names of update rules in I_j^i represent the agent actions A_j^i. The goal of each SMDP is then to find an optimal policy that maximises the reward for each visited state, according to $\pi^{*i}_j(s) = \arg\max_{a \in A_j^i \cap I_j^i} Q^{*i}_j(s,a)$, where $Q_j^{*i}(s,a)$ specifies the expected cumulative reward for executing constrained action a in state s and then following π^{*i}_j thereafter. For learning such policies we use a modified version of HSMQ-Learning. This algorithm receives subtask M_j^i and Information State I_j^i used to initialise state s, performs similarly to Q-Learning for primitive actions, but for composite actions it invokes recursively with a child subtask. In contrast to HSMQ-Learning, this algorithm chooses actions from a subset derived by applying the IS update rules to the current state of the world. When the subtask is completed, it returns a cumulative reward $r_{t+\tau}$, and continues its execution until finding a goal state for the root subtask. This process iterates until convergence occurs to optimal context-independent policies, as in HSMQ-Learning.

4 Experimental Setting

4.1 Hierarchy of Agents

Figure 2 shows a (hand-crafted) hierarchy of learning agents for navigating and acting in a situated environment. Each of these agents represents an individual generation task. Model M_0^0 is the root agent and is responsible for ensuring that a set of navigation instructions guide the user to the next referent, where an RE is generated. Model M_0^1 is responsible for the generation of the RE that best describes an intended referent. Subtasks M_0^2 ... M_2^2 realise surface forms of possible distractors, or macro- / micro landmarks. Model M_2^1 is responsible for the generation of navigation instructions which smoothly fit into the linguistic consistency pattern chosen. Part of this task is choosing between a low-level (model M_3^2) and a high-level (model M_4^2) instruction. Subtasks $M_0^3...M_4^3$ realise the actual instructions, destination, direction, orientation, path, and 'straight', respectively.[4] Finally, model M_1^1 can repair previous system utterances.

[4]Note that navigation instructions and REs correspond to sequences of actions, not to a single one.

Model(s)	Actions
M_0^0	navigation, manipulation, confirmation, stop, repair_system_act, repair_no_system_act
M_0^1	insert_distractor, insert_no_distractor, insert_no_absolute_property, insert_micro_relatum, insert_macro_relatum insert_no_taxonomic_property, insert_absolute_property, insert_no_macro_relatum, insert_taxonomic_property
M_2^1	choose_high_level, choose_low_level, get_route, choose_easy_route, choose_short_route
M_0^2... M_2^2	exp_head, exp_no_head, insert_colour, insert_no_colour, insert_size, insert_no_size, exp_spatial_relation
M_3^2	choose_explicit_abstractness, choose_implicit_abstractness, destination_instruction, path_instruction
M_4^2	choose_explicit_abstractness, choose_implicit_abstractness, direction_instr, orientation_instr, straight_instr
M_0^3... M_4^3	exp_actor, exp_no_actor, exp_mood, exp_loc_phrase, exp_no_loc_phrase, exp_process, exp_no_process

Table 2: *Action set of the learning agents and Information States.*

4.2 State and Action Sets

The HRL agent's knowledge base consists of all situational and linguistic knowledge the agent needs for decision making. Figure 2 shows the hierarchy of learning agents together with the knowledge base of the learning agent with respect to the semantic features shown in Table 1 that were used for the annotation of the GIVE-2 corpus dialogues. The first column of the table in Figure 2 indicates the respective model, also referred to as agent, or subtask, and the second column refers to the knowledge variable it uses (in the form of the feature index given in the first column of Table 1). In the agent, boolean values and strings were represented as integers. The HIS shares all information of the learning agent, but has an additional set of relational feature-value pairs for each slot. For example, if the agent knows that the slot $content(nav\_instruction)$ has value 1 (meaning 'filled'), the HIS knows also which value it was filled with, such as $path$. Such additional knowledge is required for the supervised learning baseline (Section 5). The action set of the hierarchical learning agent and the hierarchical information state is given in Table 2. The state-action space size of a flat learning agent would be $|S \times A| = 10^{11}$, the hierarchical setting has a state-action space size of 2.4×10^7. The average state-action space size of all subtasks is $|S \times A|/14 = 1.7 \times 10^7$. Generation actions can be primitive or composite. While the former correspond to single generation decisions, the latter represent separate generation subtasks (Fig. 2).

4.3 Prior Knowledge

Prior knowledge can include decisions obvious to the system designer, expert knowledge, or general intuitions. In our case, we use a supervised learning approach to induce prior knowledge into our HRL agent. We trained decision trees on our annotated corpus data using Weka's (Witten and Frank, 2005) J48 decision tree classifer. A separate tree was trained for each semantic attribute (cf. Table 1). The obtained decision trees represent our supervised learning baseline. They achieved an accuracy of 91% in a ten-fold cross-validation. For our semi-learnt combination of HRL and HIS, we performed a manual analysis of the resulting rules to assess their impact on a learning agent.[5] In the end, the following rules were used to constrain the agent's behaviour: (1) In REs, always use a referent's colour, except in cases of repair when colour is not discriminating; (2) mention a distractor or micro landmark, if the colour of the referent is not discriminating; (3) in navigation, always make orientation instructions explicit. All remaining behaviour was subject to learning.

4.4 Reward Function

We use the following reward function to train the hierarchy of policies of our HRL agent. It aims to reduce discourse length at maximal task success[6] using a consistent navigation strategy.

$$\text{R} = \begin{cases} 0 & \text{for} \quad \text{reaching the goal state} \\ \text{-2} & \text{for} \quad \text{an already invoked subtask} \\ \text{+1} & \text{for} \quad \text{generating instruction } u \text{ consistent with instruction } u_{-1} \\ \text{-1} & \quad\quad\quad \text{otherwise.} \end{cases}$$

[5]We excluded rules that always choose the same value, since they would work against our aim of generating consistent, but variable instructions.

[6]Task success is addressed by that the user has to 'accept' each instruction for a state transition.

The third reward that encourages consistency of instructions rewards a sequence of actions that allow the last generated instruction to be classified as belonging to the same navigation strategy/user model as the previously generated instruction (cf. 2.2).

5 Experiments and Results

5.1 The Simulated Environment

The simulated environment contains two kinds of uncertainties: (1) uncertainty regarding the state of the environment, and (2) uncertainty concerning the user's reaction to a system utterance. The first aspect is represented by a set of contextual variables describing the environment, [7] and user behaviour.[8] Altogether, this leads to 115 thousand different contextual configurations, which are estimated from data (cf. Section 2.1). The uncertainty regarding the user's reaction to an utterance is represented by a Naive Bayes classifier, which is passed a set of contextual features describing the situation, mapped with a set of semantic features describing the utterance.[9] From these data, the classifier specifies the most likely user reaction (after each system act) of *perform_desired_action*, *perform_undesired_action*, *wait* and *request_help*.[10] The classifier was trained on the annotated data and reached an accuracy of 82% in a ten-fold cross validation.

5.2 Learnt Policies

With respect to REs, the **fully-learnt policy** (only HRL) uses colour when it is discriminating, and a distractor or micro landmark otherwise. The **semi-learnt policy** (HRL with HIS) behaves as defined in Section 4.3. The **supervised learning policy** (only HIS) uses the rules learnt by the decision trees. Both learnt policies learn to maximise task success, and to generate consistent navigation strategies.[11] The supervised learning policy generates successful instructions from the start. Note that we are not actually learning dialogue strategies, but rather generation strategies using dialogue features. Therefore the described policies, fully-learnt, semi-learnt and supervised-learning, exclusively guide the system's behaviour in the interaction with the simulated user. An example dialogue is shown in Table 3. We can observe that the agent starts using a low level navigation strategy, and then switches to high level. When the user gets confused, the system temporarily switches back to low level. For referring expressions, it first attempts to locate the referent by reference to a distractor, and then repairs by using a micro landmark. The surface forms of instructions were realised from templates, since the NLG system so far only generates a sequence of content selection decisions. We address surface realisation in Dethlefs and Cuayáhuitl (2011).

We compared our semi-learnt policy against a fully-learnt, and a supervised learning baseline. All policies were trained for 40 thousand episodes. For training, the step-size parameter α, which indicates the learning rate, was initiated with 1 and then reduced over time by $\alpha = \frac{1}{1+t}$, where t is the time step. The discount rate γ, which indicates the relevance of future rewards in relation to immediate rewards, was set to 0.99, and the probability of a random action ϵ was 0.01. See (Sutton and Barto, 1998) for details on these parameters.

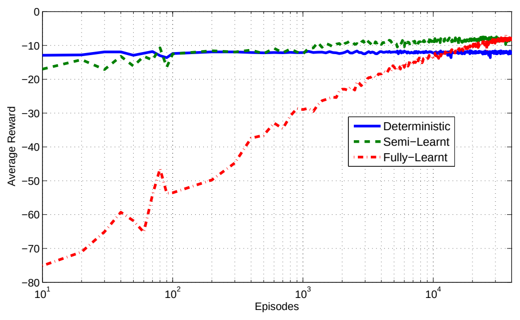


Figure 3: Comparison of fully-learnt, semi-learnt, and supervised learning (deterministic) behaviours.

5.3 Comparison of Policies

Figure 3 compares the rewards averaged over 10 runs. In terms of **learning speed**, we can see that

[7] previous system act, route length, route status (known/unknown), objects within vision, objects within dialogue history, number of instructions, alignment(proportion)

[8] previous user reaction, user position, user waiting(true/false), user type(explorative/hesitant/medium)

[9] navigation level(high / low), abstractness(implicit / explicit), repair(yes / no), instruction type(destination / direction / orientation / path / straight)

[10] User reactions measure the system's task success.

[11] They thereby also learn to adapt their semantic choices to those most frequently made by humans.

Agent	Policy	Action (composite in italics)	Utterance
Usr		*request_route*	How do I get to the trophy?
Sys	π_0^0, π_2^1	*root*, *navigation*, dontRepair, orientation	
	π_3^2, π_0^3	*low level navigation*, add direction+degrees	First, turn 90 degrees left.
	π_3^2, π_1^3	*low level navigation*, 'straight'	Then go straight ahead.
Usr			*perform_desired_action*
Sys	π_0^0	confirmation	OK.
	π_4^2, π_4^3, π_2^3	*high level navigation*, destination, direction,	
	π_4^2, π_4^3, π_2^3	addLocation	Turn left at the end of the hallway.
Usr			*wait*
Sys	π_1^1	*repair system act*, change navigation strategy	
	π_3^2, π_2^3	*low level navigation*, direction	Turn right
	π_4^2, π_3^3	*high level navigation*, path	and go down the hallway.
Usr			*perform_desired_action*
Sys	π_3^2, π_2^3	*low level navigation*, direction	Now turn left.
Usr			*perform_desired_action*
Sys	π_0^0	confirmation	OK.
	π_4^2, π_4^3	*high level navigation*, destination,	
	π_4^2, π_4^3	addLocation	Enter to the room in front of you.
Usr			*perform_desired_action*
Sys	π_0^1, π_1^2	*manipulation*, use colour, use distractor	Press the red button beside the blue.
Usr			*request_help*
Sys	π_1^1	*repair system act*, change granularity	
	π_4^2, π_0^1	*high level navigation, manipulation*	
	π_4^3, π_2^2	destination, use colour, use micro landmark	Go to the red button beside the chair.
Usr			*perform_desired_action*
Sys	π_0^1	*manipulation*	Press it.
Usr			*perform_desired_action*
Sys	π_0^0	confirmation	Well done.

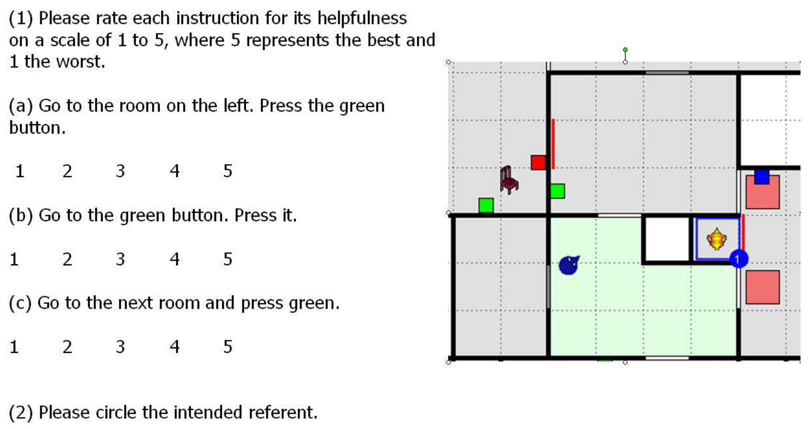
Table 3: Sample dialogue in the GIVE-2 scenario showing the dynamics of generation policies. See Figure 2 for the corresponding hierarchy models, and Table 2 for the action set. See Section 5.2 for an explantation of the dialogue.

while the semi-learnt behaviour is able to follow a near-optimal policy from the beginning, the fully-learnt policy takes about 40 thousand episodes to reach the same performance. In terms of simulated **task success**, we see that while the supervised learning behaviour follows a good policy from the start, it is eventually beaten by the learnt policies.

5.4 Human Evaluation Study

We asked 11 participants[12] to rate altogether 132 sets of instructions, where each set contained a spatial graphical scene containing a person, mapped with one human, one learnt, and one supervised learning instruction. Instructions consisted of a navigation instruction followed by a referring expression. Subjects were asked to rate instructions on a 1-5 Likert scale (where 5 is the best) for their helpfulness on guiding the displayed person from its origin to pressing the intended button. We selected six different scenarios for the evaluation: (a) only one button is present, (b) two buttons are present, the referent and a distractor of the same colour as the referent, (c) two buttons are present, the referent and a distractor of a different colour than the referent, (d) one micro landmark is present and one distractor of the same colour as the referent, (e) one micro landmark is present and one distractor of a different colour than the referent. All scenarios oc-

[12]6 female, 5 male with an age average of 26.4.


(1) Please rate each instruction for its helpfulness on a scale of 1 to 5, where 5 represents the best and 1 the worst.

(a) Go to the room on the left. Press the green button.

1 2 3 4 5

(b) Go to the green button. Press it.

1 2 3 4 5

(c) Go to the next room and press green.

1 2 3 4 5

(2) Please circle the intended referent.

Figure 4: Example scenario of the human evaluation study.

curred twice in each evaluation sheet, their specific instances were drawn from the GIVE-2 corpus at random. Scenes and instructions were presented in a randomised order. Figure 4 presents an example evaluation scene. Finally, we asked subjects to circle the object they thought was the intended referent. Subjects rated the human instructions with an average of 3.82, the learnt instructions with an average of 3.55, and the supervised learning instructions with an average of 2.39. The difference between human and learnt is not significant. The difference between learnt and supervised learning is significant at $p < 0.003$, and the difference between human and supervised learning is significant at $p < 0.0002$. In 96% of all cases, users were able to identify the intended referent.

6 Conclusion and Discussion

We have presented a combination of HRL with a hierarchical IS, which was informed by prior knowledge from decision trees. Such a combined framework has the advantage that it allows us to systematically pre-specify (obvious) generation strategies, and thereby find solutions faster, reduce computational demands, scale to complex domains, and incorporate expert knowledge. By applying HRL to the remaining (non-obvious) action set, we are able to learn a flexible, generalisable NLG policy, which will take the best action even under uncertainty. As an application of our approach and its generalisability across domains, we have presented the joint optimisation of two separate NLG tasks, navigation instructions and referring expressions, in situated dialogue under the aspects of task success and linguistic consistency. Based on an evaluation in a simulated environment estimated from data, we showed that our semi-learnt behaviour outperformed a fully-learnt baseline in terms of learning speed, and a supervised learning baseline in terms of average rewards. Human judges rated our instructions significantly better than the supervised learning instructions, and close to human quality. The study revealed a task success rate of 96%. Future work can transfer our approach to different applications to confirm its benefits, and induce the agent's reward function from data to test in a more realistic setting.

Acknowledgments

Thanks to the German Research Foundation DFG and the Transregional Collaborative Research Centre SFB/TR8 'Spatial Cognition' and the EU-FP7 project ALIZ-E (ICT-248116) for partial support of this work. Also, thanks to John Bateman for comments on an earlier draft of this paper.

References

Srinivas Bangalore and Owen Rambow. 2000. Exploiting a probabilistic hierarchical model for generation. In *Proceedings of the 18th conference on Computational linguistics - Volume 1*, pages 42–48.

Anja Belz. 2008. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering*, 1:1–26.

Peter Bohlin, Robin Cooper, Elisabet Engdahl, and Staffan Larsson. 1999. Information states and dialogue move engines. In *IJCAI-99 Workshop on Knowledge and Reasoning in Practical Dialogue Systems*.

Johan Bos, Ewan Klein, Oliver Lemon, and Tetsushi Oka. 2003. DIPPER: Description and Formalisation of an Information-State Update Dialogue System Architecture. In *4th SIGDial Workshop on Discourse and Dialogue*, pages 115–124.

Heriberto Cuayáhuitl, Steve Renals, Oliver Lemon, and Hiroshi Shimodaira. 2010. Evaluation of a hierarchical reinforcement learning spoken dialogue system. *Computer Speech and Language*, 24(2):395–429.

Heriberto Cuayáhuitl. 2009. *Hierarchical Reinforcement Learning for Spoken Dialogue Systems*. Ph.D. thesis, School of Informatics, University of Edinburgh.

Nina Dethlefs and Heriberto Cuayáhuitl. 2010. Hierarchical Reinforcement Learning for Adaptive Text Generation. *Proceedings of INLG '10*.

Nina Dethlefs and Heriberto Cuayáhuitl. 2011. Hierarchical Reinforcement Learning and Hidden Markov Models for Task-Oriented Natural Language Generation. In *Proceedings of ACL-HLT 2011, Portland, OR*.

Thomas G. Dietterich. 1999. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303.

Pablo A. Duboue and Kathleen R. McKeown. 2001. Empirically estimating order constraints for content planning in generation. In *ACL '01*, pages 172–179.

Andrew Gargett, Konstantina Garoufi, Alexander Koller, and Kristina Striegnitz. 2010. The give-2 corpus of giving instructions in virtual environments. In *LREC*.

Konstantina Garoufi and Alexander Koller. 2010. Automated planning for situated natural language generation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1573–1582, July.

Michael A. K. Halliday and Ruqaiya Hasan. 1976. *Cohesion in English*. Longman, London.

Peter Heeman. 2007. Combining reinforcement learning with information-state update rules. In *Human Technology Conference (HLT)*, pages 268–275.

Srinivasan Janarthanam and Oliver Lemon. 2010. Learning to adapt to unknown users: referring expression generation in spoken dialogue systems. In *ACL '10*, pages 69–78.

Alexander Koller and Matthew Stone. 2007. Sentence generation as planning. In *Proceedings of ACL-07*.

Alexander Koller, Kristina Striegnitz, Donna Byron, Justine Cassell, Robert Dale, Johanna Moore, and Jon Oberlander. 2010. The first challenge on generating instructions in virtual environments. In M. Theune and E. Krahmer, editors, *Empirical Methods on Natural Language Generation*, pages 337–361, Berlin/Heidelberg, Germany. Springer.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *ACL-36*, pages 704–710.

Staffan Larsson and David R. Traum. 2000. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Nat. Lang. Eng.*, 6(3-4):323–340.

Diane J. Litman, Michael S. Kearns, Satinder Singh, and Marilyn A. Walker. 2000. Automatic optimization of dialogue management. In *Proceedings of the 18th conference on Computational linguistics*, pages 502–508.

William Mann and Christian M I M Matthiessen. 1983. NIGEL: A systemic grammar for text generation. Technical report, ISI/RR-85-105.

Martin J. Pickering and Simon Garrod. 2004. Toward a mechanistc psychology of dialog. *Behavioral and Brain Sciences*, 27.

Adwait Ratnaparkhi. 2000. Trainable methods for surface natural language generation. In *Proceedings of NAACL*, pages 194–201.

Verena Rieser, Oliver Lemon, and Xingkun Liu. 2010. Optimising information presentation for spoken dialogue systems. In *ACL '10*, pages 1009–1018.

Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. 2002. Optimizing Dialogue Management with Reinforcement Learning: Experiments with the NJFun System. *Journal of Artificial Intelligence Research*, 16:105–133.

Laura Stoia, Darla Magdalene Shockley, Donna K. Byron, and Eric Fosler-Lussier. 2006. Noun phrase generation for situated dialogs. In *Proceedings of INLG '06*, pages 81–88.

Richard S Sutton and Andrew G Barto. 1998. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA.

Sebastian Varges. 2003. *Instance-based Natural Language Generation*. Ph.D. thesis, School of Informatics, University of Edinburgh.

Jette Viethen and Robert Dale. 2008. The use of spatial relations in referring expression generation. In *Proceedings of INLG '08*, INLG '08, pages 59–67.

Michael White. 2004. Reining in CCG chart realization. In *In Proc. INLG-04*, pages 182–191.

Jason Williams. 2008. The best of both worlds: Unifying conventional dialog systems and POMDPs. In *Interspeech*, Brisbane.

Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*. 2. edition.