Combining Hierarchical Reinforcement Learning and Bayesian Networks for Natural Language Generation in Situated Dialogue

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

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

Abstract

Language generators in situated domains face a number of content selection, utterance planning and surface realisation decisions, which can be strictly interdependent. We therefore propose to optimise these processes in a joint fashion using Hierarchical Reinforcement Learning. To this end, we induce a reward function for content selection and utterance planning from data using the PARADISE framework, and suggest a novel method for inducing a reward function for surface realisation from corpora. It is based on generation spaces represented as Bayesian Networks. Results in terms of task success and human-likeness suggest that our unified approach performs better than a baseline optimised in isolation or a greedy or random baseline. It receives human ratings close to human authors.

1 Introduction

Natural Language Generation (NLG) systems that work in situated domains and need to generate utterances during an interaction are faced with a number of challenges. They need to adapt their decisions to a continuously changing interaction history and spatial context as well as to the user's properties, such as their individual information needs and verbal or nonverbal responses to each generated utterance. Decisions involve the tasks of content selection, utterance planning and surface realisation, which can be in many ways related and interdependent. For the former two tasks, e.g., there is a tradeoff between how much information to include in an utterance (to increase task success), and how much a user can actually comprehend online. With regard to surface realisation, decisions are often made according to a language model of the domain (Langkilde and Knight, 1998; Bangalore and Rambow, 2000; Oh and Rudnicky, 2000; White, 2004; Belz, 2008). However, there are other linguistic phenomena, such as alignment (Pickering and Garrod, 2004), consistency (Halliday and Hasan, 1976), and variation, which influence people's assessment of discourse (Levelt and Kelter, 1982) and generated output (Belz and Reiter, 2006; Foster and Oberlander, 2006). We therefore argue that it is important to optimise content selection, utterance planning and surface realisation in a unified fashion, and we suggest to use Hierarchical Reinforcement Learning (HRL) with Bayesian networks to achieve this. Reinforcement learning (RL) is an attractive framework for optimising NLG systems, where situations are mapped to actions by maximising a long term reward signal (Rieser et al., 2010; Janarthanam and Lemon, 2010). HRL has the additional advantage of scaling to large search spaces (Dethlefs and Cuayáhuitl, 2010). Since an HRL agent will ultimately learn the behaviour it is rewarded for, the reward function is arguably the agent's most crucial component. Previous work has therefore suggested to learn a reward function from human data as in the PARADISE framework (Walker et al., 1997). We will use this framework to induce a reward function for content selection and utterance planning. However, since PARADISE relies heavily on task success metrics, it is not ideally suited for surface realisation, which depends more on linguistic phenomena like frequency, consistency and variation. Linguistic and psycho-

Proceedings of the 13th European Workshop on Natural Language Generation (ENLG), pages 110–120,
Nancy, France, September 2011. ©2011 Association for Computational Linguistics

logical studies (cited above) show that such phenomena are worth modelling in an NLG system. The contribution of this paper is therefore to induce a reward function from human data, specifically suited for surface generation. We obtain Bayesian Networks (BNs) (Jensen, 1996) from a human corpus and use them to inform the agent's learning process. We compare their performance against a greedy and a random baseline. In addition, we suggest to optimise content selection, utterance planning and surface realisation decisions in a joint, rather than isolated, fashion in order to correspond to their interrelated nature. Results in terms of task success and human-likeness show that our combined approach performs better than baselines that were optimised in isolation or act on behalf of the language model alone. Since generation spaces in our approach can be obtained for any domain for which corpus data is available, it generalises to different domains with limited effort and reduced development time.

2 Related Work

Related approaches using graphical models for NLG include Barzilay and Lee (2002) and Mairesse et al. (2010). Barzilay and Lee use multiple sequence alignment to obtain lattices of surface form variants for a semantic concept. Mairesse et al. use Dynamic Bayesian networks and learn surface form variants from semantically aligned data. Both approaches demonstrated that graphical models can yield good results for surface realisation.

Related work has also shown the benefits of treating interrelated decisions jointly. Lemon (2010) suggests to use RL to jointly optimise dialogue management and language generation for information presentation, where the system needs to learn when presentation is most advantageous. Cuayáhuitl and Dethlefs (2011b) use HRL for the joint optimisation of spatial behaviours and dialogue behaviours in an agent that learns to give route instructions by taking the user's individual prior knowledge into account. Angeli et al. (2010) treat content selection and surface realisation in a joint fashion using a log-linear classifier, which allows each decision to depend on all decisions made previously. These recent investigations show that jointly optimised policies outperform policies optimised in isolation.

3 The Generation Domain

We address the generation of navigation instructions in a virtual 3D world in the GIVE scenario (Koller et al., 2010). In this task, two people engage in a 'treasure hunt', where one participant instructs the other in navigating through the world, pressing a sequence of buttons and completing the task by obtaining a trophy. The GIVE-2 corpus (Gargett et al., 2010) provides 63 English and 45 German transcripts of such dialogues. We complemented the English dialogues with a set of semantic annotations, please see Sec. 5.1 for the knowledge base of the learning agent, which corresponds to the annotation scheme.

A key feature of the situated approach to generation we are addressing is a tight coupling of system and user behaviour as is also standard in dialogue management.[1] It allows the system to constantly monitor the user's behaviour and change its strategy as soon as the user shows signs of confusion. Since the user needs to process system utterances online, we face a tradeoff between generating few utterances (preferred by users) and generating utterances which are easy to comprehend online (increasing task success). Figure 1 contrasts the dynamics of two possible NLG system architectures, a traditional pipeline and the joint architecture suggested here. In the traditional model, an interaction always starts with information about the user, the dialogue history and the spatial setting being sent to the **content selection** (CS) component. Here, the system chooses whether to use a high-level (e.g, 'go to the next room') or a low-level navigation strategy (e.g., 'go straight, turn left'). High-level instructions are forms of contracted low-level instructions. CS also determines a level of detail for an instruction based on the number of present objects, lengths of instructions and confusion of the user. A first semantic form[2] is constructed here and passed on to **utterance planning** (UP). Here, the system decides whether to use temporal markers, conjunctions, a marked or unmarked theme as well as a mode of presentation (all together or one by one). It then

[1] In fact, some content selection decisions we treat as part of NLG here concerning the user or next system utterance may be shared with a dialogue manager in a complete dialogue system.

[2] Semantic forms contain an instruction type ('destination', 'direction', 'orientation', 'path' or 'straight'), a direction of navigation, and salient landmarks along the path of navigation.

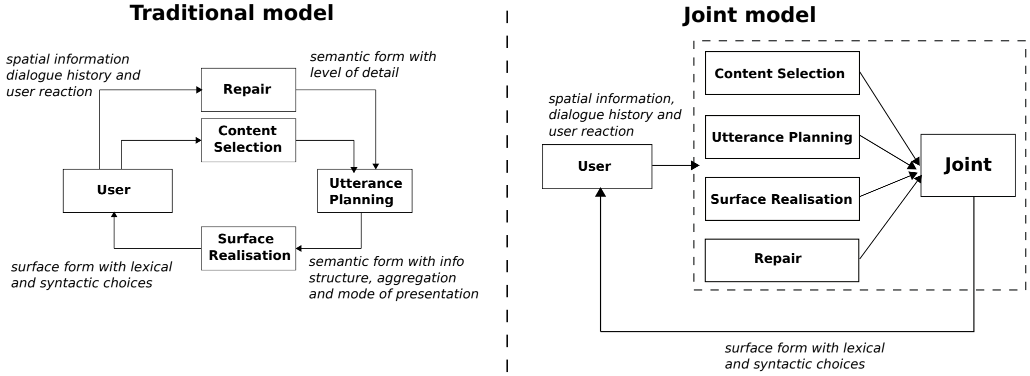

Figure 1: Left: traditional pipeline architecture of an NLG system for CS, UP and SR. Right: an architecture for joint decision making among these tasks. Information passed between components is given in cursive fonts.

consults **surface realisation** (SR) for a final realisation. The SR component addresses the one-to-many relationship between a semantic form and its possible realisations. It optimises the tradeoff between alignment and consistency (Pickering and Garrod, 2004; Halliday and Hasan, 1976) on the one hand, and variation (to improve text quality and readability) on the other (Belz and Reiter, 2006; Foster and Oberlander, 2006). The SR component produces a string of words and presents it to the user whose reaction is observed. The utterance is then either repaired (if the user hesitates or performs an undesired action) or the next one is generated. Note that CS, UP and SR are closely related in this setting. For successful CS, we may wish to be as detailed as possible in an utterance. On the other hand, redundant detail may confuse the user and make it difficult to process utterances online. In UP, we may want to generate as few utterances as possible and thus aggregate them. However, if instructions are too many, a one by one presentation may ease comprehension. In SR, a short utterance is often most likely according to a language model, but it may not be ideal when the user needs more detail. In the joint architecture, there is thus no sequential order on decision making. Instead, one best utterance is generated by considering all variables jointly across subtasks.

4 HRL with Bayesian Networks for NLG

4.1 Hierarchical Reinforcement Learning

The concept 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 the generation task (e.g. CS, UP, SR). The action set describes the system's capabilities (e.g. *'use high level navigation strategy'*, *'use imperative mood'*, 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 induce an optimal strategy. To do that we use HRL in order to optimise a hierarchy of generation policies rather than a single policy. We denote the hierarchy of RL agents as M^i_j, where the indexes i and j only identify a model in a unique way, they do not specify the execution sequence of subtasks because that is learnt. 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, and T^i_j is a probabilistic state transition function that determines the next state s' from the current state s and the performed action a. $R^i_j(s', \tau | s, a)$ is a reward function that specifies the reward that an agent receives for taking an action a in state s lasting τ time steps (Dietterich, 1999). Since actions in SMDPs may take a variable number of time steps to complete, the random variable τ represents this number of time steps. Actions can be either primitive or composite. The former yield single rewards, the latter correspond to SMDPs and yield cumulative rewards. The goal of

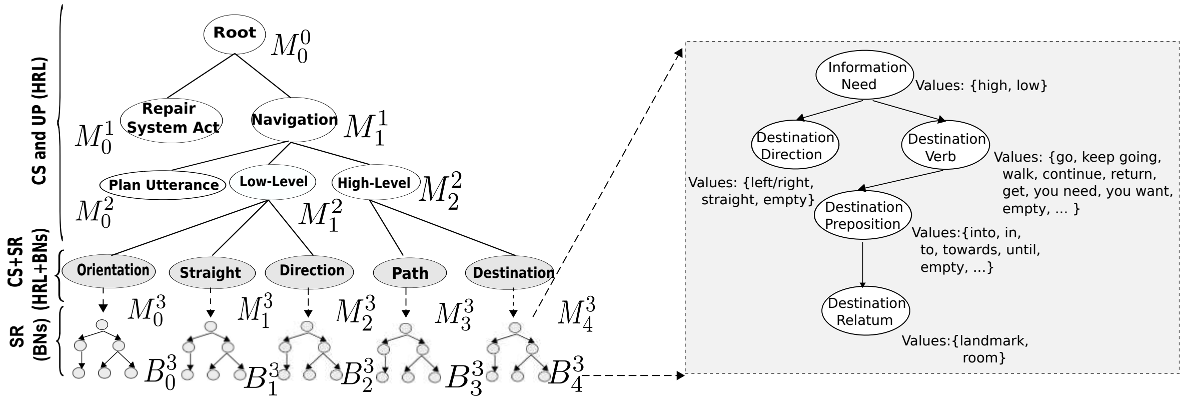

Figure 2: Hierarchy of learning agents (left). The top three layers are responsible for decisions of content selection (CS) and utterance planning (UP), and use HRL. The shaded agents in the bottom use HRL with a Bayesian Network-based reward function and joint optimisation of CS and surface realisation (SR). The BNs represent generation spaces for SR. An example BN, representing the generation space of 'destination' instructions, is shown on the right.

each SMDP is to find an optimal policy π^* that maximises the reward for each visited state, according to $\pi^{*i}_j(s) = \arg\max_{a \in A} Q^{*i}_j(s,a)$, where $Q^i_j(s,a)$ specifies the expected cumulative reward for executing action a in state s and then following π^*. For learning NLG policies, we use HSMQ-Learning, see (Cuayáhuitl, 2009), p. 92.

4.2 Bayesian Networks for Surface Realisation

We can represent a surface realiser as a BN which models the dynamics between a set of semantic concepts and their surface realisations. A BN models a joint probability distribution over a set of random variables and their dependencies based on a directed acyclic graph, where each node represents a variable Y_j with parents $pa(Y_j)$ (Jensen, 1996). Due to the Markov condition, each variable depends only on its parents, resulting in a unique joint probability distribution $p(Y) = \Pi p(Y_j|pa(Y_j))$, where every variable is associated with a conditional probability distribution $p(Y_j|pa(Y_j))$. We use random variables to represent semantic concepts and their values as corresponding surface forms. A random variable with the semantics 'destination process' e.g. can have different values 'go', 'walk', 'elided surface form' (empty) etc. The BNs were constructed manually so as to capture two main dependencies. First, the random variable 'information need' should influence the inclusion of all optional semantic constituents (on the right of Figure 2, e.g., 'destination direction') and the process of the utterance ('destination verb'). Second, a sequence of dependencies spans from the verb to the end of the utterance. In Figure 2, this is from the verb over the preposition to the relatum. The first dependency is based on the intuition that whenever the user's information need is high, optional semantic information is more likely to be included than when the information need is low.[3] Also, we assume that high frequency verb forms are preferable in cases of a high information need. The second dependency is based on the hypothesis that the value of one constituent can be estimated based on the previous constituent. In the future, we may compare different configurations and designs as well as effects of word order. Since BNs allow for probabilistic reasoning, that is the calculation of posterior probabilities given a set of query variable-value pairs, we can perform reasoning over surface forms. Given the word sequence represented by linguistic variables $Y_0...Y_n$ (lexical and syntactic information), and context and situation-based variables $Y_0...Y_m$, we can compute the posterior probability of a random variable Y_j. We use efficient implementations of the variable elimination and junction tree algorithms (Cozman, 2000) for probabilistic reasoning. Initial prior and conditional probability tables were estimated from the GIVE corpus using Maximum Likelihood Estimation.

[3] This is key to the joint treatment of CS and SR: if an utterance is not ideally informative in terms of content, it will receive bad rewards, even if good SR choices have been made (and vice versa).

5 Experimental Setting

5.1 Hierarchy of Agents: State and Action Sets

Figure 2 shows a (hand-crafted) hierarchy of learning agents for navigating and acting in a situated environment. Each agent represents an individual generation task. The models shown in the bottom of the figure represent the BNs $B_0^3 ... B_4^3$ that inform SR decisions. The state representation contains all situational and linguistic knowledge the agent requires for optimal decision making. The following are the state and action sets of the agents in Figure 2 (see the corresponding feature structures). Model M_0^0 is the root agent, it decides whether to generate the next instruction, repair a previous utterance (M_0^1), or confirm the user's behaviour. Model M_1^1 is responsible for navigation instruction generation.[4] It has information about the situational context (e.g., visible objects, route length), the status of the utterance, and the user. It chooses a navigation level, and an utterance plan.[5] State variable names can be reused in later agents. The value 'filled' means that a decision has been made, 'unfilled' means it is still open. Model M_0^2 performs UP. It makes decisions concerning aggregation, info structure, temporal markers and utterance presentation. Decisions are based on the user's information need, and the number of instructions, and do not exclude each other. Model M_1^2 generates low level instructions (direction, orientation, 'straight') based on the user's information need and waiting behaviour. Model M_2^2 generates high-level instructions (destination, path). Model M_0^3 is responsible for orientation instructions. It chooses surface forms for semantic constituents based on the user's information need and behaviour. State variables correspond to semantic concepts, their values to realisation variants. Similarly, model M_1^3 generates 'straight', and model M_2^3 direction instructions. They represent low-level navigation. Model M_3^3 generates path, and model M_4^3 destination instructions. They realise high-level navigation. The hierarchical agent has $|S \times A| = \sum_{i,j} |S_j^i| \times |A_j^i| = 2.5$ million state-action pairs.

S_1^1
v1:GoalVisible ← 0=true,1=false
v2:InformationNeed ← 0=low,1=high
v3:NavigationLevel ← 0=unfilled,1=filled
v4:PreviousUserReaction ← 0=none,1=perform action, 2=perform undesired action,3=wait, 4=request help
v5:RepairStatus ← 0=unfilled,1=filled
v6:RouteLength ← 0=short,1=long
v7:RouteStatus ← 0=unfilled,1=filled
v8:UserPosition ← 0=on track,1=off track
v9:UserWaits ← 0=true,1=false
v10:UtterancePlan ← 0=short,1=long

A_1^1
fetchRoute(), dontRepair(),
useHighLevelPlan(), useLowLevelPlan(),
repairUtterance(), generateHighLevel(),
planUtterance(), generateLowLevel(),

S_0^2
v11:Aggregation ← 0=unfilled,1=filled
v12:InfoStructure ← 0=unfilled,1=filled
v13:NumInstructions ← 1=1,2=2,3=3 or more
v14:Presentation ← 0=unfilled,1=filled
v15:TemporalMarker ← 0=unfilled,1=filled, v4, v8

A_0^2
aggregate(), dontAggregate(), temporalMarkers(),
noTemporalMarkers(), markedTheme(), unmarked-,
Theme(), jointPresentation(), incrementalPresent.

S_1^2
v16:LowLevelContent ← 0=direction,1=orientation, 2=straight; v2, v4, v8, v9
v17:NavigationAbstractness ← 0=unfilled,1=filled

A_1^2
explicitUtterance(), implicitUtterance(),
generateDirection(), generateOrientation()
generateStraight()

S_2^2
v18:HighLevelContent ← 0=destination,1=path
v2, v4, v8, v9, v17

A_2^2
explicitUtterance(), implicitUtterance(),
generateDestination(), generatePath()

S_0^3
v19:Degrees ← 0=empty,1=filled
v20:Destination ← 0=empty,1=filled
v21:Direction ← 0=empty,1=filled
v22:AddInfo ← 0=path,1=destination,2=empty 3=direction,4=orientation,5=location
v23:Verb ← 0=turn,1=keep going,2=look..., v8

A_0^3
insert turn, insert direction, insert path, etc.
(all 14 surface form variants and combinations)

S_1^3
v24:Direction ← 0=straight,1=forward,2=ahead, ...
v25:Verb ← 0=walk,1=go,2=continue..., v8, v22

A_1^3
insert go, insert straight, insert orientation, etc.
(all 11 surface form variants and combinations)

S_2^3
v26:Preposition ← 0=to(your),1=to(the),2=empty...
v27:Verb ← 0=turn,1=go,2=bear..., v8, v21, v22

A_2^3
insert go, insert to(your), insert direction, etc.
(all 12 surface form variants and combinations)

S_3^3
v28:Preposition ← 0=down,1=along,2=through,...
v29:Verb ← 0=walk,1=go,2=follow...
v30:Relatum ← 0=tunnel,1=space,2=point..., v8, v22

A_3^3
insert go, insert through, insert tunnel, etc.
(all 13 surface form variants and combinations)

S_4^3
v31:Preposition ← 0=into,1=towards,2=until,...
v32:Verb ← 0=walk,1=go,2=return..., v8, v21, v22
v33:Relatum ← 0=room,1=point,2=empty,

A_4^3
insert go, insert to, insert point, etc.
(all 12 surface form variants and combinations)

[4] Models M_0^0 and M_0^1 are omitted, since we focus on the right branch of the hierarchy in this paper, i.e. from M_1^1 down.

[5] Bold-face (composite) actions pass control between agents. Each time an agent is called, it takes between 7 and 10 (composite or primitive) actions, the exact number varies per agent.

5.2 A Reward Function for CS and UP

According to the PARADISE framework (Walker et al., 2000), the performance of a system can be modelled as a weighted function of task success and dialogue cost measures (e.g., number of turns, interaction time). We argue that PARADISE is also useful to assess the performance of an NLG system. To identify the strongest predictors of user satisfaction (US) in situated dialogue/NLG systems, we performed an analysis of subjective and objective dialogue metrics based on PARADISE. In a human evaluation study in a real setting (Dethlefs et al., 2010), 26 participants were asked to interact with a route-giving dialogue system and follow the system's instructions. Subsequently, participants provided subjective ratings of the system's performance to indicate their US. The study revealed that users prefer short interactions at maximal task success. We also found that task success metrics that penalise the degree of task difficulty correlate higher with US than binary (success/failure) metrics.[6] We therefore define graded task success (GTS) by assigning a value of 1 for finding the target location (FTL) without problems, $2/3$ for FTL with small problems and 0 for FTL with severe problems. The value with small problems was assigned for short confusions of the user, the value for severe problems was assigned if the user got lost at least once. More specifically, in order to identify the relative contribution that different factors have on the variance found in US scores, we performed a standard multiple regression analysis on the data. First results showed that 'user turns' (UT) and 'graded task success' (GTS) (which are negatively correlated) were the only predictors. In a second multiple regression analysis involving only these metrics we obtained the performance function $Performance = 0.38\mathcal{N}(GTS) - 0.87\mathcal{N}(UT)$, where 0.38 is a weight on the normalised value of GTS and 0.87 is a weight on the normalised value of UT. This result is significant at $p < 0.01$ and accounts for 62% of the variation found in US. Using this reward function (and -1 for each other action), the agent is rewarded for short interactions (few user turns) at maximal (graded) task success. User turns correspond to the behaviour with which a user reacts to an utterance. If the user reacts positively (carries out the instructions), task success is rated with 1; if they hesitate, it is $2/3$ and if they get lost (carry out a wrong instruction), it is 0. In this way the agent receives the highest rewards for the shortest possible utterance followed by a positive user reaction. This reward function is used by all CS and UP agents $M_0^0 \dots M_2^2$. Rewards are assigned after each system instruction presented to the user and the user's reaction. This reward is propagated back to all agents that contributed to the sequence of decisions leading to the instruction.

5.3 A Reward Function for Surface Realisation

Due to its unique function in an RL framework, we suggest to induce a reward function for SR from human data. To this end, we use BNs to provide feedback to an agent learning to optimise SR decisions. Whenever the agent has generated a word sequence (and reaches a goal state), it receives $P(w_0 ... w_n)$ as a reward. This corresponds to $\sum P(Y_j = v_x | pa(Y_j) = v_y)$, the sum of posterior probabilities given the chosen values v_x and v_y of random variables and their dependencies. It receives a reward of $+1$ for maintaining an equal distribution of alignment and variation. In this way, the agent learns to balance the most likely surface forms against the benefits of variation and nonlinguistic context. [7] The agent receives a reward of -1 for any other action (to encourage efficiency). Agents $M_{0...4}^3$ use this reward function.

6 Experiments and Results

6.1 The Simulated Environment

The simulated environment has two parts: simulating the spatial context of an utterance and simulating the user's reaction to it. The first part was designed using unigrams modeling features of the context [8] and the user. [9] This lead to 23 thousand dif-

[6] Graded metrics show a high correlation with user satisfaction, binary metrics only show a moderate correlation.

[7] The distribution of alignment and variation is measured by dividing the number of surface variants used before by the total number of variants used. The agent is then rewarded for keeping the resulting number around 0.5, i.e. for a middle way between alignment and variation (Dethlefs and Cuayáhuitl, 2010).

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

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

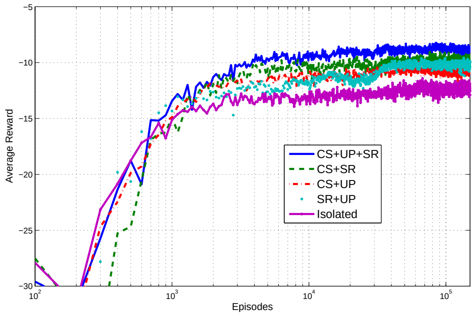

Figure 3: Performance of navigation instruction generation policies, jointly optimised and in isolation. See explanation in Section 6.2 and sample dialogue in Table 2.

Compared Instructions	F-Measure	KL-Divergence
Real1 - Real2	0.58	1.77
Real - ‘HRL with BNs’	0.38	2.83
Real - ‘HRL with *greedy*’	0.49	4.34
Real - ‘HRL with *random*’	0.0	10.06

Table 1: Evaluation of generation behaviours with Precision-Recall and KL-divergence.

ferent configurations which we estimated from the GIVE corpus to ensure the system is trained under multiple circumstances. Since the corpus contains three different worlds, we estimated the training environment from worlds 1 and 2, and the test environment from world 3. We addressed the simulation of user reactions with a Naive Bayes Classifier. It is passed a set of features describing the current context and user and a set of semantic features describing the generated utterance.[10] Based on this, the classifier returns the most likely user reaction of *perform_desired_action, perform_undesired_action, wait* and *request_help*. It reached 82% of accuracy in a 10-fold cross validation. Simulating user reactions helps to assess the quality of instructions and provides feedback to the agent’s learning process.

6.2 Comparison of Learnt Policies

We have made two main claims in this paper: (1) that CS, UP and SR decisions should all be learnt in a joint fashion to achieve optimal performance, and (2) that BNs can prove beneficial for learning SR variants. To address the first claim, Figure 3 shows the performance (in terms of average rewards)[11] of our agent with (a) isolated optimisation of CS, UP and SR, (b) joint optimisation of CS and SR, (c) joint optimisation of CS and UP, (d) joint optimisation of SR and UP and (e) joint optimisation of all subtasks. All policies were trained[12] for 150 thousand episodes, where one episode corresponds to one generated utterance. We can see that learning a joint policy for all three subtasks achieves the best performance. In terms of **content selection**, the agent learns to prefer high level navigation strategies, which allow more efficient instruction giving, and switch to low level whenever the user gets confused. Regarding **utterance planning**, the agent prefers incremental displays for three or more instructions, and joint presentations otherwise. For **surface realisation**, the agent learns to choose a (short) most likely surface form when the user has a low information need, but include more information otherwise. It learns to balance variation and alignment in an about equal proportion. Trained in isolation, a non-optimal behaviour is learnt. The reason is that all three components have a repertoire of actions, which are different in nature, but can have similar effects. For example, assume that for a user with medium information need the CS component makes a decision favouring an efficient instruction giving. It chooses a high-level navigation strategy, which contracts several low-level instructions. The next component, UP, should now take an action to balance the earlier efficiency decision and correspond to the user’s increased cognitive load. However, without access to the earlier decision, it may itself make an efficiency choice, and thus increase the likelihood of the user hesitating or requesting help.

The second claim concerning the advantage of BNs for SR is addressed by Table 1. Here, we tested the human-likeness of SR decisions by com-

[10] navigation level(high / low), repair(yes / no), instruction type(destination / direction / orientation / path / straight), aggregation(yes / no), info structure(marked / unmarked), presentation(joint / incremental), temporal markers(yes / no)

[11] Since the reward function assigns a reward of -1 for each action taken, rewards stay in negative values.

[12] For training, the step-size parameter α (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.

Conv.	Policy	Action (composite in italics)	Utterance
USR		*request_route (low info need, on track)*	*‘How do I get to the trophy?’*
	π_0^0	**CS:** *navigation*, dontConfirm	
	π_1^1	*generateHighLevel*, *planUtterance*, dontRepair	
	π_2^2	*generateDestination*, *generateDirection*	
	π_0^2	**UP:** jointPresentation, noTempMarkers	
	π_2^3	**SR:** turnVP, emptyPP, insertLocation	
	π_4^3	emptyVP, emptyPP, pointRelatum	Turn left at the end of the hall.
USR			*[waits]*
SYS	π_0^0	**CS:** *navigation*,	
	π_1^1	*generateLowLevel, planUtterance, repairUtterance*	
	π_0^1	switchNavigationStrategy	
	π_1^2	*generateDirection, generatePath*	
	π_0^2	**UP:** aggregateClauses, incrementalPresentation	
	π_2^3	**SR:** turnVP, emptyPP, noLocation	Turn right,
	π_3^3	goVP, downPrep, pathRelatum	and go down the hallway.
USR			*[executes navigation instructions]*
SYS	π_0^0	**CS:** *navigation, dontConfirm*	
	π_1^1	*generateLowLevel, planUtterance, dontRepair*	
	π_1^2	*generateDirection*	
	π_0^2	**UP:** incrementalPresentation, tempMarkers	
	π_2^3	**SR:** bearVP, emptyPP	Now bear left.
USR			*[executes navigation instructions]*
SYS	π_0^0	confirmation	Well done.

Table 2: Sample dialogue for the jointly learnt policy. See Section 5.1 for corresponding policies and actions. The agent starts using a high level navigation strategy. When the user gets confused, it temporarily switches back to low level; nonverbal behaviour is given in square brackets.

paring them with the human-authored instructions from the GIVE corpus. We compare our jointly learnt policy (‘HRL with BNs’) with a greedy baseline (‘HRL with *greedy*’, where SR decisions are made purely based on frequency) and a random baseline (‘HRL with *random*’ where SR decisions only aim to produce a grammatical form). For the comparison we use Precision-Recall based on the F-Measure score, and dialogue similarity based on the Kullback-Leibler (KL) divergence (Cuayáhuitl et al., 2005), which computes the difference between two probability distributions. In all cases, we compared word strings of human-authored instructions against word strings of instructions generated by each of our policies. Table 1 shows results of the comparison of two human data sets ‘Real1’ vs ‘Real2’ and both of them together, as the gold standard, against our policies. While ‘HRL with *greedy*’ receives higher F-Measure scores, the learnt policy is more similar to the human data. This is due to variation: in contrast to the greedy baseline, which always exploits the most likely variant, the ‘HRL with BNs’ policy learns to vary surface forms. This leads to lower F-Measure scores, but achieves higher similarity with human authors. This ultimately is a desirable property, since it enhances the quality and naturalness of our instructions. The ‘HRL with *random*’ baseline, which generates random variation, performs worst according to both metrics.

Table 2 shows an example dialogue using the jointly learnt policy. Here, the user has prior knowledge of the environment and a low information need. The system decides to give instructions efficiently and chooses a high-level navigation strategy (‘Turn left at the end of the hall’). When the user hesitates (waits), it repairs its previous utterance, switches the navigation strategy to low-level and rephrases the utterance (‘Turn right, and go to the end of the hall.

Conv.	Utterance
USR	*'How do I get to the trophy?'*
SYS	Go to the end of the hall and then turn left.
USR	*[waits]*
SYS	Turn right, go straight and follow the corridor. Then turn left.
USR	*[turns right, requests help]*
SYS	Go to the end of the hall and turn left.
USR	*[undesired action: turns left]*
SYS	Wait! Go back, then straight and then follow the corridor.
USR	*[executes instructions]*
SYS	Very good. Turn left.
USR	*[executes instructions]*
SYS	Very good.

Table 3: Sample dialogue using policy optimised in isolation. In terms of CS and UP, the agent prefers efficient instruction giving. There is no variation in surface forms.

Now bear left'). It also chose to aggregate the messages using the conjunction 'and' (to minimise the number of instructions), and present them in a one-by-one fashion (to ease comprehensibility). This interrelated decision making is possible due to their joint optimisation. In contrast, Table 3 shows a dialogue for the same situation using the policy optimised in isolation, where the user gets confused several times. Since decision making is not interrelated, all components prefer efficiency decisions (a high-level navigation strategy, aggregation and joint presentation whenever possible). There is no variation in surface forms, and repair strategies affect only the immediately preceding utterance.

6.3 Human Evaluation Study

To get a more reliable idea of the quality and human acceptance of our instructions, we asked 12 participants[13] to rate 96 sets of instructions. Each set contained a spatial graphical scene with a person, mapped with one human, one jointly learnt, and one instruction learnt in isolation. Participants were asked to rate navigation instructions to an object, e.g. 'go left and press the yellow button', on a 1-5 Likert scale (where 5 is the best) for their helpfulness on guiding the displayed person to the referent. Scenes were presented in a random order. We then asked the participants to circle the object they thought was the intended referent. Human instructions were rated with a mean of 3.86 (with a standard deviation (SD) of 0.89). The jointly learnt instructions were rated with a mean of 3.57 (SD=1.07) and instructions learnt in isolation with a mean of 2.35 (SD=0.85). The difference between human and jointly learnt is not significant ($p < 0.29$) according to a t-test. The effect size r is 0.14. The difference between human and learnt in isolation is significant at $p < 0.001$ with an effect size r of 0.65 and the difference between jointly learnt and learnt in isolation is significant at $p < 0.003$ and has an effect size r of 0.53. Users were able to identify the intended referent in 96% of all cases.

7 Conclusion

We have presented a novel approach to optimising NLG for situated interactions using HRL with BNs. We also suggested to jointly optimise the tasks of CS, UP and SR using reward functions induced from human data. For the former two, we used the PARADISE framework to obtain a reward function that favours short interactions at maximal task success. We then proposed a method for inducing a reward function for SR from human data: it uses BNs to represent the surface realiser and inform the HRL agent's learning process. In this way, we are able to address a number of challenges arising with situated NLG and correspond to the interrelated nature of different NLG tasks. Results showed that our jointly learnt policies outperform policies learnt in isolation and received human ratings similar to human instructions. We also found that our hybrid approach to SR using HRL with BNs generates language more similar to human data than a greedy or random baseline enhancing language quality and naturalness. Future work can transfer our approach to different domains, or address the effects of SR variants on human ratings in a more detailed study. Other graphical models, e.g. Dynamic Bayesian Networks, can be explored for SR. In addition, adaptive NLG during an interaction can be explored assuming a continuously changing learning environment, as shown for situated dialogue management by Cuayáhuitl and Dethlefs (2011a).

[13] 7 female, 5 male with an age average of 25.6.

Acknowledgements Thanks to the German Research Foundation DFG, the SFB/TR8 ‘Spatial Cognition’ and the EU-FP7 project ALIZ-E (ICT-248116) for partial support of this work.

References

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

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

Regina Barzilay and Lillian Lee. 2002. Bootstrapping lexical choice via multiple-sequence alignment. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 164–171.

Anja Belz and Ehud Reiter. 2006. Comparing Automatic and Human Evaluation of NLG Systems. In *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 313–320.

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

Fabio G. Cozman. 2000. Generalizing variable elimination in Bayesian networks. In *IBERAMIA/SBIA, Workshop on Probabilistic Reasoning in Artificial Intelligence*, pages 27–32, Sao Paulo, Brazil.

Heriberto Cuayáhuitl and Nina Dethlefs. 2011a. Optimizing Situated Dialogue Management in Unknown Environments. In *Proceedings of INTERSPEECH*, Florence, Italy.

Heriberto Cuayáhuitl and Nina Dethlefs. 2011b. Spatially-aware dialogue control using hierarchical reinforcement learning. *ACM Transactions on Speech and Language Processing (Special Issue on Machine Learning for Robust and Adaptive Spoken Dialogue Systems*, 7(3).

Heriberto Cuayáhuitl, Steve Renals, Oliver Lemon, and Hiroshi Shimodaira. 2005. Human-Computer Dialogue Simulation Using Hidden Markov Models. In *Proceedings of the Automatic Speech Recognition and Understanding Workshop ASRU*, pages 290–295.

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. *Proceeding of the 6th International Conference on Natural Language Generation (INLG)*.

Nina Dethlefs, Heriberto Cuayáhuitl, Kai-Florian Richter, Elena Andonova, and John Bateman. 2010. Evaluating task success in a dialogue system for indoor navigation. In *Proceedings of the Workshop on the Semantics and Pragmatics of Dialogue (SemDial), Poznan, Poland*.

Thomas G. Dietterich. 1999. Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. *Journal of Artificial Intelligence Research*, 13:227–303.

Mary Ellen Foster and Jon Oberlander. 2006. Data-driven generation of emphatic facial displays. In *Proceedings of the European Chapter of the Association for Computational Linguistic (EACL)*, pages 353–360.

Andrew Gargett, Konstantina Garoufi, Alexander Koller, and Kristina Striegnitz. 2010. The GIVE-2 corpus of giving instructions in virtual environments. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC)*.

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

Srinivasan Janarthanam and Oliver Lemon. 2010. Learning to adapt to unknown users: referring expression generation in spoken dialogue systems. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 69–78.

Finn V. Jensen. 1996. *An Introduction to Bayesian Networks*. Springer Verlag, New York.

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 *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 704–710.

Oliver Lemon. 2010. Learning what to say and how to say it: joint optimization of spoken dialogue management and natural language generation. *Computer Speech and Language*, 25(2).

Willem J. M. Levelt and S Kelter. 1982. Surface form and memory in question answering. *Cognitive Psychology*, 14.

François Mairesse, Milica Gašić, Filip Jurčíček, Simon Keizer, Blaise Thomson, Kai Yu, and Steve Young.

2010. Phrase-based statistical language generation using graphical models and active learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1552–1561.

Alice H. Oh and Alexander I. Rudnicky. 2000. Stochastic language generation for spoken dialogue systems. In *Proceedings of the 2000 ANLP/NAACL Workshop on Conversational systems - Volume 3*, pages 27–32.

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

Verena Rieser, Oliver Lemon, and Xingkun Liu. 2010. Optimising information presentation for spoken dialogue systems. In *Proceedings of the Annual Meeting of the Association for Computational Lingustics (ACL)*, pages 1009–1018.

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

Marilyn A. Walker, Diane J. Litman, Candace A. Kamm, and Alicia Abella. 1997. PARADISE: A framework for evaluating spoken dialogue agents. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 271–280.

Marilyn Walker, Candice Kamm, and Diane Litman. 2000. Towards developing general models of usability with PARADISE. *Natural Language Engineering*, 6(3):363–377.

Michael White. 2004. Reining in CCG chart realization. In *Proceedings of the International Conference on Natural Language Generation (INLG)*, pages 182–191.