Hierarchical Reinforcement Learning for Adaptive Text Generation

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

Heriberto Cuayáhuitl
University of Bremen, Germany
heriberto@uni-bremen.de

Abstract

We present a novel approach to natural language generation (NLG) that applies hierarchical reinforcement learning to text generation in the wayfinding domain. Our approach aims to optimise the integration of NLG tasks that are inherently different in nature, such as decisions of content selection, text structure, user modelling, referring expression generation (REG), and surface realisation. It also aims to capture existing interdependencies between these areas. We apply hierarchical reinforcement learning to learn a generation policy that captures these interdependencies, and that can be transferred to other NLG tasks. Our experimental results—in a simulated environment—show that the learnt wayfinding policy outperforms a baseline policy that takes reasonable actions but without optimization.

1 Introduction

Automatic text generation involves a number of subtasks. (Reiter and Dale, 1997) list the following as core tasks of a complete NLG system: content selection, discourse planning, sentence planning, sentence aggregation, lexicalisation, referring expression generation and linguistic realisation. However, decisions made for each of these core tasks are not independent of each other. The value of one generation task can change the conditions of others, as evidenced by studies in corpus linguistics, and it can therefore be undesirable to treat them all as isolated modules. In this paper, we focus on interrelated decision making in the areas of content selection, choice of text structure, referring expression and surface form. Concretely, we generate route instructions that are tailored specifically towards different user types as well as different environmental features. In addition, we aim to balance the degree of variation and alignment in texts and produce lexical and syntactic patterns of co-occurrence that resemble those of human texts of the same domain. Evidence for the importance of this is provided by (Halliday and Hasan, 1976) who note the way that lexical cohesive ties contribute to text coherence as well as by the theory of interactive alignment. According to (Pickering and Garrod, 2004) we would expect significant traces of lexical and syntactic self-alignment in texts.

Approaches to NLG in the past have been either rule-based (Reiter and Dale, 1997) or statistical (Langkilde and Knight, 1998). However, the former relies on a large number of hand-crafted rules, which makes it infeasible for controlling a large number of interrelated variables. The latter typically requires training on a large corpus of the domain. While these approaches may be better suitable for larger domains, for limited domains such as our own, we propose to overcome these drawbacks by applying Reinforcement Learning (RL)—with a hierarchical approach. Previous work that has used RL for NLG includes (Janarthanam and Lemon, 2009) who employed it for alignment of referring expressions based on user models. Also, (Lemon, 2008; Rieser and Lemon, 2009) used RL for optimising information presentation styles for search results. While both approaches displayed significant effects of adaptation, they focused on a single area of optimisation. For larger problems, however, such as the one we are aiming to solve, flat RL will not be applicable due to the large state space. We therefore sug-

gest to divide the problem into a number of subproblems and apply hierarchical reinforcement learning (HRL) (Barto and Mahadevan, 2003) to solve it.

We describe our problem in more detail in Section 2, our proposed HRL architecture in Sections 3 and 4 and present some results in Section 5. We show that our learnt policies outperform a baseline that does not adapt to contextual features.

2 Generation tasks

Our experiments are all drawn from an indoor navigation dialogue system which provides users with route instructions in a university building and is described in (Cuayáhuitl et al., 2010). We aim to optimise generation within the areas of *content selection*, *text structure*, *referring expression generation* and *surface realisation*.

Content Selection Content selection decisions are subject to different user models. We distinguish users who are familiar with the navigation environment and users who are not. In this way, we can provide different routes for these users corresponding to their particular information need. Specifically, we provide more detail for unfamiliar than familiar users by adding any or several of the following: (a) landmarks at decision points, (b) landmarks lying on long route segments, (c) specifications of distance.

Text Structure Depending on the type of user and the length of the route, we choose among three different text generation strategies to ease the cognitive load of the user. Examples of all strategies are displayed in Table 1. All three types resulted from an analysis of a corpus of 24 human-written driving route instructions. We consider the first type (sequential) most appropriate for long or medium-long routes and both types of user. The second type (temporal) is appropriate for unfamiliar users and routes of short or medium length. It divides the route into an explicit sequence of consecutive actions. The third type (schematic) is used in the remaining cases.

Referring Expression Generation We distinguish three types of referring expressions: *common names*, *familiar names* and *descriptions*. In this way, entities can be named according to the users' prior knowledge. For example, one and the same room can be called either 'the student union room', 'room A3530' or 'the room right at the corner beside the entrance to the terrace'.

Surface Realisation For surface realisation, we aim to generate texts that display a natural balance of (self-)alignment and variation. While it is a rule of writing that texts should typically contain variation of surface forms in order not to appear repetitive and stylistically poor, there is evidence that humans also get influenced by self-alignment processes during language production. Specifically, (Garrod and Anderson, 1987; Pickering and Garrod, 2004) argue that the same mental representations are used during language production and comprehension, so that alignment occurs regardless of whether the last utterance was made by another person or by the speaker him- or herself (for experimental evidence see (Branigan et al., 2000; Bock, 1986)). We can therefore hypothesise that coherent texts will, besides variation, also display a certain degree of self-alignment. In order to determine a proper balance of alignment and variation, we computed the degree of lexical repetition from our corpus of 24 human route descriptions. This analysis was based on (Hirst and St-Onge, 1998) who retrieve lexical chains from texts by identifying a number of relations between lexical items. We focus here exclusively on Hirst & St-Onge's 'extra-strong' relations, since these can be computed from shallow properties of texts and do not require a large corpus of the target domain. In order to make a fair comparison between the human texts and our own, we used a part-of-speech (POS) tagger (Toutanova and Manning, 2000)[1] to extract those grammatical categories that we aim to control within our framework, i.e. nouns, verbs, prepositions, adjectives and adverbs. Based on these categories, we compute the proportion of tokens that are members in lexical chains, the 'alignment score' (AS), according to the following equation:

$$AS = \frac{\text{Lexical tokens in chains}}{\text{Total number of tokens}} \times 100. \quad (1)$$

We obtained an average alignment score of 43.3% for 24 human route instructions. In contrast, the

[1] http://nlp.stanford.edu/software/tagger.shtml

Table 1: *Different text generation strategies for the same underlying route.*

Type 1: Sequential	Type 2: Temporal	Type 3: Schematic
Turn around, and go straight to the glass door in front of you. Turn right, then follow the corridor until the lift. It will be on your left-hand side.	First, turn around. Second, go straight to the glass door in front of you. Third, turn right. Fourth, follow the corridor until the lift. It will be on your left-hand side.	- Turn around. - Go straight until the glass door in front of you. (20 m) - Turn right - Follow the corridor until the lift. (20 m) - It will be on your left-hand side.

same number of instructions generated by Google Maps yielded 78.7%, i.e. an almost double amount of repetition. We will therefore train our agent to generate texts with an about medium alignment score.

3 Hierarchical Reinforcement Learning for NLG

The idea of *text generation as an optimization problem* is as follows: given a set of generation states, a set of actions, and an objective reward function, an optimal generation strategy maximizes 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. content selection, text structure, REG, surface realization). The action set describes the system's capabilities (e.g. expand_sequential_aggregation, expand_schematic_aggregation, expand lexical items, etc.). The reward function assigns a numeric value for each taken action. In this way, text 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 that we use hierarchical reinforcement learning in order to optimize a hierarchy of text generation policies rather than a single policy.

The hierarchy of RL agents consists of L levels and N models per level, denoted as $M = M_j^i$, 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 $< S_j^i, A_j^i, T_j^i, R_j^i >$. S_j^i is a set of states, A_j^i is a set of actions, and T_j^i is a transition function that determines the next state s' from the current state s and the performed action a with a probability of $P(s'|s,a)$. $R_j^i(s', \tau|s, a)$ is a reward function that specifies the reward that an agent receives for taking an action a in state s at time τ. Since SMDPs allow for temporal abstraction, that is, actions 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 (executed using a stack mechanism) correspond to SMDPs and yield cumulative discounted rewards. The goal of each SMDP is to find an optional policy π^* that maximises the reward for each visited state, according to

$$\pi^{*i}_j(s) = \arg\max_{a \in A} Q^{*i}_j(s, a). \quad (2)$$

where $Q_j^i(s, a)$ specifies the expected cumulative reward for executing action a in state s and then following π^*. For learning a generation policy, we use hierarchical Q-Learning (HSMQ) (Dietterich, 1999). The dynamics of SMDPs are as follows: when an SMDP terminates its execution, it is popped off the stack of models to execute, and control is transferred to the next available SMDP in the stack, and so on until popping off the root SMDP. An SMDP terminates when it reaches one of its terminal states. This algorithm is executed until the Q-values of the root agent stabilize. The hierarchical decomposition allows to find context-independent policies with the advantages of policy reuse and facilitation for state-action abstraction. This hierarchical approach has been applied successfully to dialogue strategy learning (Cuayahuitl et al., 2010).

4 Experimental Setting

4.1 Hierarchy of SMDPs

The hierarchy consists of 15 agents. It is depicted in Figure 1. The root agent is responsible for deter-

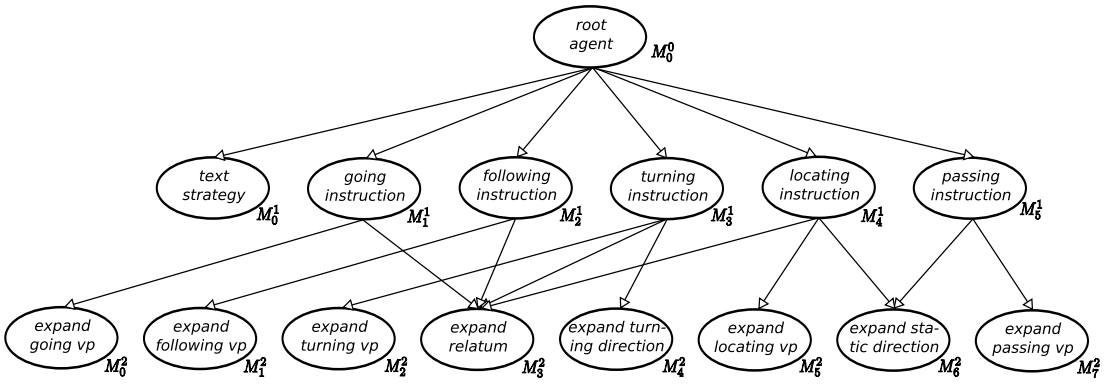

Figure 1: Hierarchy of agents for learning adaptive text generation strategies in the wayfinding domain

mining a route instruction type for a navigation situation. We distinguish turning, passing, locating, going and following instructions. It also chooses a text generation strategy and the information structure of the clause (i.e., marked or unmarked theme (Halliday and Matthiessen, 2004)). Leaf agents are responsible for expanding constituents in which variation or alignment can occur, e.g. the choice of verb or prepositional phrase.

4.2 State and action sets

We distinguish three kinds of state representations, displayed in Table 2. The first (M_{10}^0 and M_0^1) encodes information on the spatial environment and user type so that texts can be tailored towards these variables. These variables play a major part in our simulated environment (Section 5.1). The second representation (M_1^1 - M_5^1 and M_3^2) controls sentence structure and ensures that all required constituents for a message have been realised. The third (all remaining models) encodes variants of linguistic surface structure and represents the degree of alignment of all variants. We address the way that these alignment values are computed in Section 4.4. Actions can be either primitive or composite. Whereas the former expand a logical form directly, the latter correspond to SMDPs at different levels of the hierarchy. All parent agents have both types of actions, only the leaf agents have exclusively primitive actions. The set of primitive actions is displayed in Table 2, all composite actions, corresponding to models, are shown in Figure 1. The average number of state-action pairs for a model is $|S \times A| = 77786$. While in the present work, the action set was determined manually, future work can aim at learning hierarchies of SMDPs automatically from data.

4.3 Prior Knowledge

Agents contain prior knowledge of two sorts. First, the root agents and agents at the first level of the hierarchy contain prior probabilities of executing certain actions. For example, given an unfamiliar user and a long route, model M_0^1, text_strategy, is initiated with a higher probability of choosing a sequential text strategy than a schematic or temporal strategy. Second, leaf agents of the hierarchy are initiated with values of a hand-crafted language model. These values indicate the probabilities of occurrence of the different surface forms of the leaf agents listed in Table 2. Both types of prior probabilities are used by the reward functions described below.

4.4 Reward functions

We use two types of reward function, both of which are directly motivated by the principles we stated in Section 2. The first addresses interaction length (the shorter the better) and the choice of actions tailored towards the user model and spatial environment.

$$\mathrm{R} = \begin{cases} 0 & \text{for reaching the goal state} \\ \text{-10} & \text{for an already invoked subtask} \\ p(a) & \text{otherwise} \end{cases} \quad (3)$$

$p(a)$ corresponds to the probability of the last action given the current state, described above as prior knowledge. The second reward function addresses

Table 2: *State and action sets for learning adaptive text generation strategies in the wayfinding domain*

Model	State Variables	Action Set
M_0^0	text_strategy (FV), info_structure (FV), instruction (FV), slot_in_focus(0=action, 1=landmark), user_type(0=unfamiliar, 1=familiar) subtask_termination(0=continue, 1=halt)	expand_text_strategy (M_0^1), turning (M_3^2), going (M_1^2), passing (M_5^2), following (M_2^2), locating_instr.(M_4^2), expand_unmarked_theme
M_0^1	end(0=continue, 1=halt), text_strategy (FV), route_length (0=short, 1=medium, 2=long), user_type(0=unfam., 1=fam.)	expand_schematic_aggregation, expand_sequence_aggregation, expand_temporal_aggregation
M_1^1	going_vp (FV), limit (FV), SV	expand_going_vp (M_0^2), expand_limit
M_2^1	following_vp (FV), SV, limit (FV)	expand_following_vp (M_1^2), expand_limit
M_3^1	turning_location (FV), turning_vp (FV), SV, turning_direction (FV)	expand_turning_vp (M_2^3), expand_turning_loc., expand_turning_direction (M_4^3)
M_4^1	np_locatum (FV), locating_vp (FV), static_direction (FV), SV	expand_np_locatum, expand_locating_vp (M_5^2), expand_static_dir. (M_6^2)
M_5^1	np_locatum (FV), passing_vp (FV), SV, static_direction (FV)	expand_pass._vp (M_6^2), expand_static_dir. (M_6^2)
M_0^2	vp_go_straight_ahead, vp_go_straight, vp_move_straight_ahead, vp_walk_straight_ahead, vp_walk_straight (all AS)	Actions correspond to expansions of lexemes
M_1^2	vp_follow, vp_go_over, vp_walk_down, vp_go_down, vp_go_up, vp_walk_up, vp_walk_over (all AS)	Actions correspond to expansions of lexemes
M_2^2	vp_walk, vp_veer , vp_hang, vp_bear (all AS), vp_go, vp_head, vp_turn (all AS)	Actions correspond to expansions of lexemes
M_3^2	identifiability(0=not id.,1=id.), user_type(0=un-, fam.,1=fam,, relatum_identifiability (FV), relatum_name (FV)	expand_relatum_id., expand_relatum, _not_id., expand_descriptive, expand_common_name
M_4^2	pp_nonphoric, pp_nonphoric_handedness, pp_nonphoric_poss, pp_phoric pp_nonphoric_side (all AS)	Actions correspond to expansions of lexemes
M_5^2	vp_be, vp_be_located_at, vp_get_to, vp_see (all AS)	Actions correspond to expansions of lexemes
M_6^2	direction_on, direction_poss, direction_to (all AS)	Actions correspond to expansions of lexemes
M_7^2	vp_move_past, vp_pass, vp_pass_by, vp_walk_past (all AS)	Actions correspond to expansions of lexemes

(FV = filling status): 0=unfilled, 1=filled. (SV = shared variables): the variables np_actor (FV), relatum (FV), sentence (FV) and information_need (0=low, 1=high) are shared by several subagents; the same applies to their corresponding expansion actions. (AS = alignment score): 0=unaligned, 1=low AS, 2=medium AS, 3=high AS.

the tradeoff between alignment and variation:

$$\mathrm{R} = \begin{cases} 0 & \text{for} \quad \text{reaching the goal state} \\ p(a) & \text{for} \quad \text{medium alignment} \\ \text{-0.1} & \quad \text{otherwise} \end{cases} \tag{4}$$

Whilst the former reward function is used by the root and models M_0^1 - M_5^1 and M_2^2, the latter is used by models M_0^2 - M_1^2 and M_3^2 - M_7^2. It rewards the agent for a medium alignment score, which corresponds to the score of typical human texts we computed in Section 2. The alignment status of a constituent is computed by the Constituent Alignment Score (CAS) as follows, where MA stands for 'medium alignment'.

$$CAS(a) = \frac{\text{Count of occurrences}(a)}{\text{Occurences of } a \text{ without MA}} \tag{5}$$

From this score, we can determine the degree of alignment of a constituent by assigning 'no alignment' for a constituent with a score of less than 0.25, 'low alignment' for a score between 0.25 and 0.5, 'medium alignment' for a score between 0.5 and 0.75 and 'high alignment' above. On the whole thus, the agent's task consists of finding a balance between choosing the most probable action given the language model and choosing an action that aligns with previous utterances.

5 Experiments and Results

5.1 Simulated Environment

The simulated environment encodes information on the current user type (un-/familiar with the environment) and corresponding information need (low or high), the length of the current route (short, medium-long, long), the next action to perform (turn, go straight, follow a path, pass a landmark or take note of a salient landmark) and the current focus of attention (the action to be performed or some salient landmark nearby). Thus, there are five different state variables with altogether 120 combinations, sampled from a uniform distribution. This simple form of stochastic behaviour is used in our simulated environment. Future work can consider inducing a learning environment from data.

5.2 Comparison of learnt and baseline policies

In order to test our framework, we designed a simulated environment that simulates different navigational situations, routes of different lengths and different user types. We trained our HRL agent for 10.000 episodes with the following learning parameters: the step-size parameter α was initiated with 1 and then reduced over time by $\alpha = \frac{1}{1+t}$, t being the time step. The discount rate parameter γ was 0.99 and the probability of random action ϵ was 0.01 (see (Sutton and Barto, 1998) for details on these parameters). Figure 2 compares the learnt behaviour of our agent with a baseline (averaged over 10 runs) that chooses actions at random in models M_0^1 and M_0^2 - M_7^2 (i.e., the baseline does not adapt its text strategy to user type or route length and neither performs adaptation of referring expressions or alignment score). The user study reported in (Cuayáhuitl et al., 2010) provided users with instruction using this baseline generation behaviour. The fact that users had a user satisfaction score of 90% indicates that this is a sensible baseline, producing intelligible instructions. We can observe that after a certain number of episodes, the performance of the trained agent begins to stabilise and it consistently outperforms the baseline.

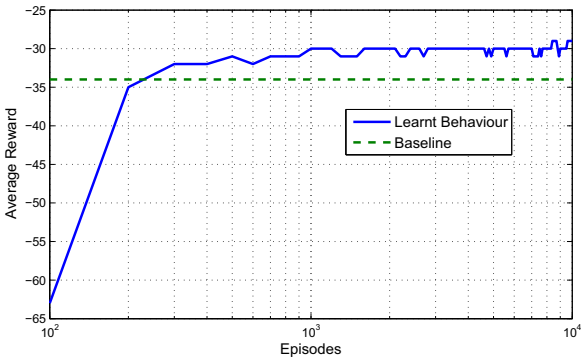

Figure 2: Comparison of learnt and baseline behaviour in the generation of route descriptions

6 Example of generation

As an example, Figure 3 shows in detail the generation steps involved in producing the clause 'Follow the corridor until the copyroom' for an unfamiliar user and a route of medium length. Generation starts with the root agent in state (0,0,0,0,0,0), which indicates that text_strategy, info_structure and instruction are unfilled slots, the slot_in_focus of the sentence is an action, the status of subtask_termination is 'continue' and the user_type is unfamiliar. After the primitive action expand_unmarked_theme was executed, the state is updated to (0,1,0,0,0,0), indicating the filled slot. Next, the composite action text_strategy is executed, corresponding to model M_0^1. The initial state (1,0,0) indicates a route of medium length, an unfilled text_strategy slot and an unfamiliar user. After the primitive action expand_sequential_text was chosen, the terminal state is reached and control is returned to the root agent. Here, the next action is following_instruction corresponding to model M_2^1. The initial state (0,1,0,0,0,0) here indicates unfilled slots for following_vp, np_actor, sentence, path, limit and relatum, as well as a high information_need of the current user. The required constituents are expanded in turn. First, the primitive actions expand_limit, expand_np_actor, expand_s and expand_path cause their respective slots in the state representation to be filled. Next, the composite action expand_relatum is executed with an initial state (0,1,0,0) representing an identifiable landmark, unfilled slots for a determiner and a referring expression for the landmark and an unfamiliar user. Two primitive actions, expand_relatum_identifiable and expand_relatum_common_name, cause the agent to

reach its terminal state. The generated referring expression thus treats the referenced entity as either known or easily recoverable. Finally, model M_1^2 executes the composite action expand_following_vp, which is initialised with a number of variables corresponding to the alignment status of different verb forms. Since this is the first time this agent is called, none of them shows traces of alignment (i.e., all values are 0). Execution of the primitive action expand_following_vp causes the respective slot to be updated and the agent to terminate. After this subtask, model M_2^1 has also reached its terminal state and control is returned to the root agent.

As a final step towards surface generation, all chosen actions are transformed into an SPL (Kasper, 1989). The type 'following instruction' leads to the initialisation of a semantically underspecified scaffold of an SPL, all other actions serve to supplement this scaffold to preselect specific syntactic structures or lexical items. For example, the choice of 'expand_following_vp' leads to the lexical item 'follow' being inserted. Similarly, the choice of 'expand_path' leads to the insertion of 'the corridor' into the SPL to indicate the path the user should follow. 'expand_limit', in combination with the choice of referring expression, leads to the insertion of the PP 'until the copy room'. For generation of more than one instruction, aggregation has to take place. This is done by iterating over all instructions of a text and inserting them into a larger SPL that realises the aggregation. Finally, the constructed SPL is passed to the KPML surface generator (Bateman, 1997) for string realisation.

7 Discussion

We have argued in this paper that HRL is an especially suited framework for generating texts that are adaptive to different users, to environmental features and properties of surface realisation such as alignment and variation. While the former tasks appear intuitively likely to contribute to users' comprehension of texts, it is often not recognised that the latter task can have the same effect. Differing surface forms of identical concepts in texts without motivation can lead to user confusion and deteriorate task success. This is supported by Clark's 'principle of contrast' (Clark, 1987), according to which new expressions are only introduced into an interaction when the speaker wishes to contrast them with other entities already present in the discourse. Similiarly, a study by (Clark and Wilkes-Gibbs, 1986) showed that interlocutors tend to align their referring expressions and thereby achieve more efficient and successful dialogues. We tackled the integration of different NLG tasks by applying HRL and presented results, which showed to be promising. As an alternative to RL, other machine learning approaches may be conceivable. However, supervised learning requires a large amount of training data, which may not always be available, and may also produce unpredictable behaviour in cases where a user deviates from the behaviour covered by the corpus (Levin et al., 2000). Both arguments are directly transferable to NLG. If an agent is able to act only on grounds of what it has observed in a training corpus, it will not be able to react flexibly to new state representations. Moreover, it has been argued that a corpus for NLG cannot be regarded as an equivalent gold standard to the ones of other domains of NLP (Belz and Reiter, 2006; Scott and Moore, 2006; Viethen and Dale, 2006). The fact that an expression for a semantic concept does not appear in a corpus does not mean that it is an unsuited or impossible expression. Another alternative to pure RL is to apply semi-learnt behaviour, which can be helpful for tasks with very large state-action spaces. In this way, the state-action space is reduced to only sensible state-action pairs by providing the agent with prior knowledge of the domain. All remaining behaviour continues to be learnt. (Cuayáhuitl, 2009) suggests such an approach for learning dialogue strategies, but again the principle is transferable to NLG. While there is room for exploration of different RL methods, it is clear that neither traditional rule-based accounts of generation, nor n-gram-based generators can achieve the same flexible generation behaviour given a large, and partially unknown, number of state variables. Since state spaces are typically very large, specifying rules for each single condition is at best impractical. Especially for tasks such as achieving a balanced alignment score, as we have shown in this paper, decisions depend on very fine-grained textual cues such as patterns of co-occurrence which are hard to pin down accurately by hand. On the other hand, statistical approaches

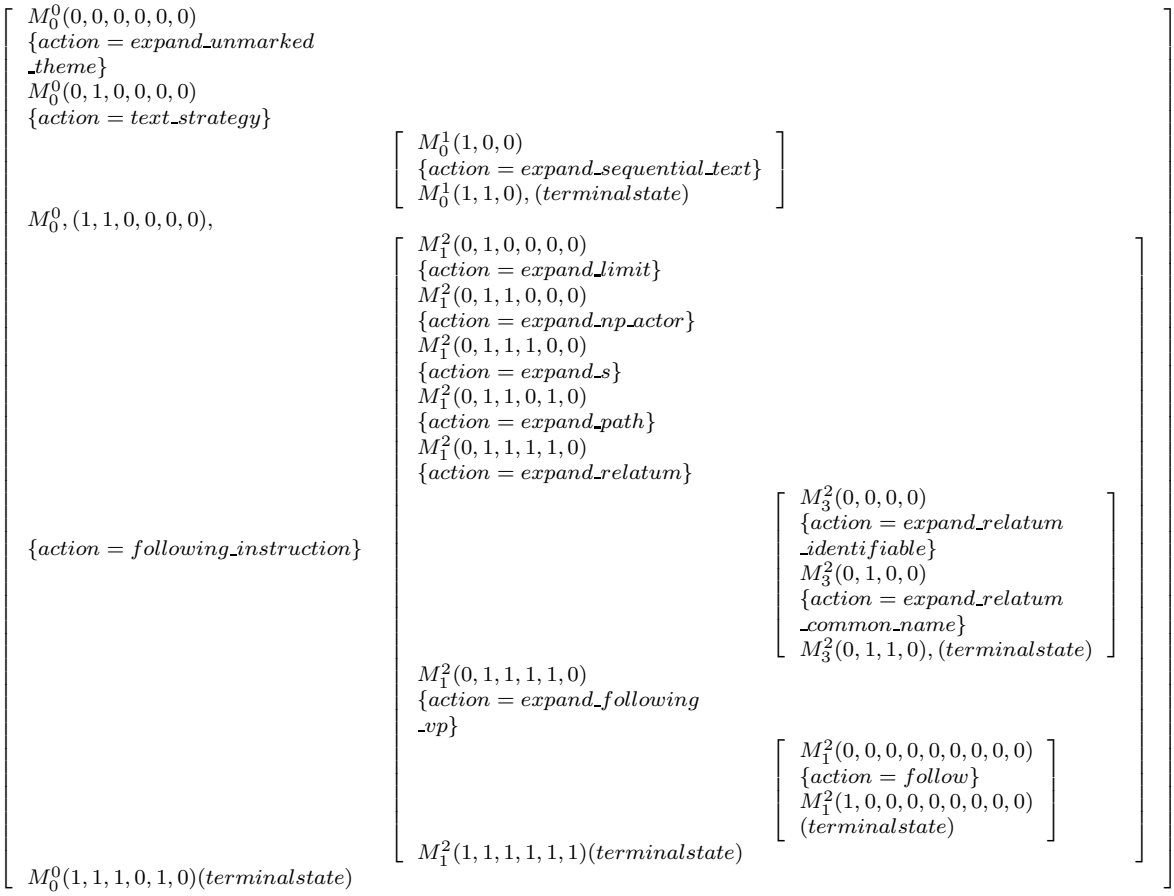

Figure 3: *Example of generation for the clause 'Follow the corridor until the copy room'. This example shows decision making for a single instruction, adaptation and alignment occurs over longer sequences of text.*

to generation that are based on n-grams focus on the frequency of constructions in a corpus without taking contextual variables such as user type or environmental properties into account. Further, they share the problem of supervised learning approaches discussed above, namely, that it can act only on grounds of what it has observed in the past, and are not well able to adapt to novel situations. For a more detailed account of statistical and trainable approaches to NLG as well as their advantages and drawbacks, see (Lemon, 2008).

8 Conclusion

We presented a novel approach to text generation that applies hierarchical reinforcement learning to optimise the following interrelated NLG tasks: content selection, choice of text structure, referring expressions and surface structure. Generation decisions in these areas were learnt based on three different variables: the type of user, the properties of the spatial environment and the proportion of alignment and variation in texts. Based on a simulated environment, we compared the results of different policies and demonstrated that the learnt policy outperforms a baseline that chooses actions without taking contextual variables into account. Future work can transfer our approach to different domains of application or to other NLG tasks. In addition, our preliminary simulation results should be confirmed in an evaluation study with real users.

Acknowledgements

This work was partly supported by DFG SFB/TR8 "Spatial Cognition".

References

Barto, A. G. and Mahadevan, S. (2003). Recent Advances in Hierarchical Reinforcement Learning. *Discrete Event Dynamic Systems*, 13:2003.

Bateman, J. A. (1997). Enabling technology for multilingual natural language generation: the KPML development environment. *Natural Language Engineering*, 3(1):15–55.

Belz, A. and Reiter, E. (2006). Comparing automatic and human evaluation of nlg systems. In *In Proc. EACL06*, pages 313–320.

Bock, K. (1986). Syntactic persistence in language production. *Cognitive Psychology*, 18.

Branigan, H. P., Pickering, M. J., and Cleland, A. (2000). Syntactic coordination in dialogue. *Cognition*, 75.

Clark, E. (1987). The principle of contrast: A constraint on language acquisition. In MacWhinney, B., editor, *Mechanisms of Language Acquisition*, pages 1–33. Lawrence Erlbaum Assoc., Hillsdale, NJ.

Clark, H. H. and Wilkes-Gibbs, D. (1986). Referring as a colloborative process. *Cognition*, 22.

Cuayáhuitl, H. (2009). *Hierarchical Reinforcement Learning for Spoken Dialogue Systems*. PhD thesis, School of Informatics, University of Edinburgh.

Cuayáhuitl, H., Dethlefs, N., Richter, K.-F., Tenbrink, T., and Bateman, J. (2010). A dialogue system for indoor wayfinding using text-based natural language. *International Journal of Computational Linguistics and Applications, ISSN 0976-0962*.

Cuayahuitl, H., Renals, S., Lemon, O., and Shimodaira, H. (2010). Evaluation of a hierarchical reinforcement learning spoken dialogue system. *Computer Speech and Language*, 24(2):395–429.

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

Garrod, S. and Anderson, A. (1987). Saying What You Mean in Dialogue: A Study in conceptual and semantic co-ordination. *Cognition*, 27.

Halliday, M. A. K. and Hasan, R. (1976). *Cohesion in English*. Longman, London.

Halliday, M. A. K. and Matthiessen, C. M. I. M. (2004). *An Introduction to Functional Grammar*. Edward Arnold, London, 3rd edition.

Hirst, G. and St-Onge, D. (1998). Lexical chains as representations of context for the detection and correction of malapropisms. In Fellbaum, C., editor, *WordNet: An Electronic Database and Some of its Applications*, pages 305–332. MIT Press.

Janarthanam, S. and Lemon, O. (2009). Learning lexical alignment policies for generating referring expressions in spoken dialogue systems. In *ENLG '09: Proceedings of the 12th European Workshop on Natural Language Generation*, pages 74–81, Morristown, NJ, USA.

Kasper, R. (1989). SPL: A Sentence Plan Language for text generation. Technical report, USC/ISI.

Langkilde, I. and Knight, K. (1998). Generation that exploits corpus-based statistical knowledge. In *ACL-36: Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 704–710.

Lemon, O. (2008). Adaptive Natural Language Generation in Dialogue using Reinforcement Learning. In *SemDial*.

Levin, E., Pieraccini, R., and Eckert, W. (2000). A stochastic model of computer-human interaction for learning dialogue strategies. *IEEE Transactions on Speech and Audio Processing*, 8.

Pickering, M. J. and Garrod, S. (2004). Toward a mechanistc psychology of dialog. *Behavioral and Brain Sciences*, 27.

Reiter, E. and Dale, R. (1997). Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.

Rieser, V. and Lemon, O. (2009). Natural language generation as planning under uncertainty for spoken dialogue systems. In *EACL '09: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 683–691, Morristown, NJ, USA.

Scott, D. and Moore, J. (2006). An NLG evaluation competition? eight reasons to be cautious. Technical report.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA.

Toutanova, K. and Manning, C. D. (2000). Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora*, pages 63–70, Morristown, NJ, USA. Association for Computational Linguistics.

Viethen, J. and Dale, R. (2006). Towards the evaluation of referring expression generation. In *In Proceedings of the 4th Australiasian Language Technology Workshop*, pages 115–122.