Optimising Incremental Generation for Spoken Dialogue Systems: Reducing the Need for Fillers

Nina Dethlefs, Helen Hastie, Verena Rieser and Oliver Lemon
Heriot Watt University
EH14 4AS, Edinburgh
n.s.dethlefs | h.hastie | v.t.rieser | o.lemon@hw.ac.uk

Abstract

Recent studies have shown that incremental systems are perceived as more reactive, natural, and easier to use than non-incremental systems. However, previous work on incremental NLG has not employed recent advances in statistical optimisation using machine learning. This paper combines the two approaches, showing how the *update*, *revoke* and *purge* operations typically used in incremental approaches can be implemented as state transitions in a Markov Decision Process. We design a model of incremental NLG that generates output based on micro-turn interpretations of the user's utterances and is able to optimise its decisions using statistical machine learning. We present a proof-of-concept study in the domain of Information Presentation (IP), where a learning agent faces the trade-off of whether to present information as soon as it is available (for high reactiveness) or else to wait until input ASR hypotheses are more reliable. Results show that the agent learns to avoid long waiting times, fillers and self-corrections, by re-ordering content based on its confidence.

1 Introduction

Traditionally, the smallest unit of speech processing for interactive systems has been a full utterance with strict, rigid turn-taking. Components of these interactive systems, including NLG systems, have so far treated the utterance as the smallest processing unit that triggers a module into action. More recently, work on incremental systems has shown that processing smaller 'chunks' of user input can improve the user experience (Skantze and Schlangen, 2009; Buss et al., 2010; Skantze and Hjalmarsson, 2010; Baumann et al., 2011). Incrementality in NLG systems enables the system designer to model several dialogue phenomena that play a vital role in human discourse (Levelt, 1989) but have so far been absent from NLG systems. These include more natural turn-taking through rapid system responses, grounding through the generation of backchannels and feedback, and barge-ins (from both user and system). In addition, corrections and self-corrections through constant monitoring of user and system utterances play an important role, enabling the system to recover smoothly from a recognition error or a change in the user's preferences. Some examples of the phenomena we are targeting are given in Fig. 1.

Skantze and Hjalmarsson (2010) present a model of incremental speech generation in which input processing and output planning are parallel processes and the system can self-monitor its own generation process. In an evaluation with human users they showed that their incremental system started to speak significantly faster than a non-incremental system (roughly 600 ms) and was perceived as significantly more polite and efficient. Users also indicated that they knew better when to start speaking themselves. Alternative approaches to incremental NLG include Kilger and Finkler (1995) who present an early approach based on Tree-Adjoining Grammar, and Purver and Otsuka (2003) who define an incremental generator based on Dynamic Syntax. Both of these generators can monitor their own output and initiate corrections if necessary.

Over recent years, adaptive and data-driven ap-

Self-correction (the system made a mistake)
USR I want Italian food in the centre of town ...
SYS OK. I found 35 Indian restaurants ...
USR No, I want Italian.
SYS oh sorry ...
SYS I have 24 Italian restaurants in the city centre ...

Correction (the user changed their mind)
USR I want Indian food in the centre of town ...
SYS There are 35 Indian ...
USR Oh sorry, I meant Italian.
SYS OK, Italian ...
SYS I have 24 Italian restaurants ...

Re-ordering (from high to low confidence)
USR I want Italian food ...
SYS [waits]
USR in the city centre.
SYS I have 120 places in the city centre ...
USR I love Italian ...
SYS 24 of them Italian ...

Holding the floor
USR I want cheap Italian food ...
SYS ok let me see
SYS I have 3 cheap Italian places ...

Figure 1: Example phenomena generated with the trained policy. The agent has learnt to produce backchannels when the user pauses, monitor and (self-)correct its output, and present information according to its confidence.

proaches to NLG have also been developed and shown to outperform the previous (handcrafted, rule-based) methods for specific problems (Rieser et al., 2010; Janarthanam and Lemon, 2010; Dethlefs and Cuayáhuitl, 2011). This work has established that NLG can fruitfully be treated as a data-driven statistical planning process, where the objective is to maximise expected utility of the generated utterances (van Deemter, 2009), by adapting them to the context and user. Statistical approaches to sentence planning and surface realisation have also been explored (Stent et al., 2004; Belz, 2008; Mairesse et al., 2010; Angeli et al., 2010). The advantages of data-driven methods are that NLG is more robust in the face of noise, can adapt to various contexts and, trained on real data, can produce more natural and desirable variation in system utterances.

This paper describes an initial investigation into a novel NLG architecture that combines incremental processing with statistical optimisation. In order to move away from conventional strict-turn taking, we have to be able to model the complex interactions observed in human-human conversation. Doing this in a deterministic fashion through hand-written rules would be time consuming and potentially inaccurate, with no guarantee of optimality. In this paper, we demonstrate that it is possible to learn incremental generation behaviour in a reward-driven fashion.

2 Previous Work: Incremental Processing Architectures

The smallest unit of processing in incremental systems is called *incremental unit* (IU). Its instantiation depends on the particular processing module. In speech recognition, IUs can correspond to phoneme sequences that are mapped onto words (Baumann and Schlangen, 2011). In dialogue management, IUs can correspond to dialogue acts (Buss et al., 2010). In speech synthesis, IUs can correspond to speech unit sequences which are mapped to segments and speech plans (Skantze and Hjalmarsson, 2010). IUs are typically linked to other IUs by two types of relations: *same-level* links connect IUs sequentially and express relationships at the same level; *grounded-in* links express hierarchical relations between IUs.

2.1 Buffer-Based Incremental Processing

A general abstract model of incremental processing based on buffers and a processor was developed by Schlangen and Skantze (2009) and is illustrated in Figure 2. It assumes that the *left buffer* of a module, such as the NLG module, receives IUs from one or more other processing modules, such as the dialogue manager. These input IUs are then passed on to the *processor*, where they are mapped to corresponding (higher-level) IUs. For an NLG module, this could be a mapping from the dialogue act *present(cuisine=Indian)* to the realisation *'they serve Indian food'*. The resulting IUs are passed on to the *right buffer* which co-incides with the left buffer of another module (for example the speech synthesis module in our example). Same-level links are indicated as dashed arrows in Figure 2 and grounded-in links as stacked boxes of IUs.

The figure also shows that the mapping between IUs can be a one-to-many mapping (IU1 and IU2 are mapped to IU3) or a one-to-one mapping (IU3 is

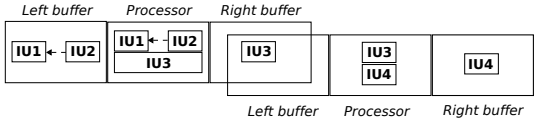

Figure 2: The buffer-based model showing two connected modules (from Skantze and Hjalmarsson (2010).

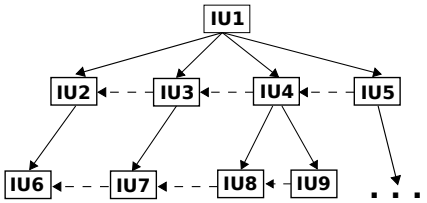

Figure 3: The ISU-model for incremental processing (adapted from Buss and Schlangen (2011)).

mapped to IU4). The model distinguishes four operations that handle information processing: *update, revise, purge* and *commit*. Whenever new IUs enter the module's left buffer, the module's knowledge base is *updated* to reflect the new information. Such information typically corresponds to the current best hypothesis of a preceding processing module. As a property of incremental systems, however, such hypotheses can be *revised* by the respective preceding module and, as a result, the knowledge bases of all subsequent modules need to be *purged* and *updated* to the newest hypothesis. Once a hypothesis is certain to not be revised anymore, it is *committed*. For concrete implementations of this model, see Skantze and Schlangen (2009), Skantze and Hjalmarsson (2010), Baumann and Schlangen (2011).

An implementation of an incremental dialogue manager is based on the Information State Update (ISU) model (Buss et al., 2010; Buss and Schlangen, 2011). The model is related in spirit to the buffer-based architecture, but all of its input processing and output planning is realised by ISU rules. This is true for the incremental 'house-keeping' actions update, revise, etc. and all types of dialogue acts. The incremental ISU model is shown in Figure 3. Note that this hierarchical architecture transfers well to the "classical" division of NLG levels into utterance (IU1), content selection (IU2 - IU5) and surface realisations (IU6 - IU9, etc.).

2.2 Beat-Driven Incremental Processing

In contrast to the buffer-based architectures, alternative incremental systems do not reuse previous partial hypotheses of the user's input (or the system's best output) but recompute them at each processing step. We follow Baumann et al. (2011) in calling them *'beat-driven'* systems. Raux and Eskenazi (2009) use a cost matrix and decision theoretic principles to optimise turn-taking in a dialogue system under the constraint that users prefer no gaps and no overlap at turn boundaries. DeVault et al. (2009) use maximum entropy classification to support responsive overlap in an incremental system by predicting the completions of user utterances.

2.3 Decision-making in Incremental Systems

Some of the main advantages of the buffer- and ISU-based approaches include their inherently incremental mechanisms for updating and revising system hypotheses. They are able to process input of varying size and type and, at the same time, produce arbitrarily complex output which is monitored and can be modified at any time. On the other hand, current models are based on deterministic decision making and thus share some of the same drawbacks that non-incremental systems have faced: (1) they rely on hand-written rules which are time-consuming and expensive to produce, (2) they do not provide a mechanism to deal with uncertainty introduced by varying user behaviour, and (3) they are unable to generalise and adapt flexibly to unseen situations.

For NLG in particular, we have seen that incrementality can enhance the responsiveness of systems and facilitate turn-taking. However, this advantage was mainly gained by the system producing semantically empty fillers such as *um, let me see, well*, etc. (Skantze and Hjalmarsson, 2010). It is an open research question whether such markers of planning or turn-holding can help NLG systems, but for now it seems that they could be reduced to a minimum by optimising the *timing and order of Information Presentation*. In the following, we develop a model for incremental NLG that is based on reinforcement learning (RL). It learns the best moment to present information to the user, when faced with the options of presenting information as soon as it becomes available or else waiting until the in-

Type	Example
Comparison	The restaurant *Roma* is in the medium price range, but does not have great food. The *Firenze* and *Verona* both have great food but are more expensive. The *Verona* has good service, too.
Recommendation	Restaurant *Verona* has the best overall match with your query. It is a bit more expensive, but has great food and service.
Summary	I have 43 Italian restaurants in the city centre that match your query. 10 of them are in the medium price range, 5 are cheap and 8 are expensive.

Table 1: Examples of IP as a *comparison*, *recommendation* and *summary* for a user looking for Italian restaurants in the city centre that have a good price for value.

put hypotheses of the system are more stable. This also addresses the general trade-off that exists in incremental systems between the processing speed of a system and the output quality.

3 Information Presentation Strategies

Our domain of application will be the Information Presentation phase in an interactive system for restaurant recommendations, extending previous work by Rieser et al. (2010), (see also Walker et al. (2004) for an alternative approach). Rieser et al. incrementally construct IP strategies according to the predicted user reaction, whereas our approach focuses on timing and re-ordering of information according to dynamically changing input hypotheses. We therefore implement a simplified version of Rieser et al.'s model. Their system distinguished two steps: the selection of an IP strategy and the selection of attributes to present to the user. We assume here that the choice of attributes is determined by matching the types specified in the user input, so that our system only needs to choose a strategy for presenting its results (in the future, though, we will include attribute selection into the decision process). Attributes include *cuisine*, *food quality*, *location*, *price range* and *service quality* of a restaurant. The system then performs a database lookup and chooses among three main IP strategies *summary*, *comparison*, *recommendation* and several ordered combinations of these. Please see Rieser et al. (2010) for details. Table 1 shows examples of the main types of presentation strategies we address.

4 Optimising Incremental NLG

To optimise the NLG process within an incremental model of dialogue processing, we define an RL agent with incremental states and actions for the IP task. An RL agent is formalised as a Markov Decision Process, or MDP, which is characterised as a four-tuple $< S, A, T, R >$, where S is a set of states representing the status of the NLG system and all information available to it, A is a set of NLG actions that combine strategies for IP with handling incremental updates in the system, T is a probabilistic transition function that determines the next state s' from the current state s and the action a according to a conditional probability distribution $P(s'|s, a)$, and R is a reward function that specifies the reward (a numeric value) that an agent receives for taking action a in state s. Using such an MDP, the NLG process 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 t is the time step. Note that a learning episode falls naturally into a number of time steps at each of which the agent observes the current state of the environment s_t, takes an action a_t and makes a transition to state s_{t+1}. This organisation into discrete time steps, and the notion of a state space that is accessible to the learning agent at any time allows us to implement the state *update*, *revoke* and *purge* operations typically assumed by incremental approaches as state updates and transitions in an MDP. Any change in the environment, such as a new best hypothesis of the recogniser, can thus be represented as a transition from one state to another. At each time step, the agent then takes the currently best action according to the new state. The best action in an incremental framework can include *correcting* a previous output, *holding the floor* as a marker of planning, or to *wait* until presenting information.[1]

[1]We treat these actions as part of NLG content selection here, but are aware that in alternative approaches, they could

States
incrementalStatus {0=none,1=holdFloor,2=correct,3=selfCorrect}
presStrategy {0=unfilled,1=filled}
statusCuisine {0=unfilled,1=low,2=medium,3=high,4=realised}
statusFood {0=unfilled,1=low,2=medium,3=high,4=realised}
statusLocation {0=unfilled,1=low,2=medium,3=high,4=realised}
statusPrice {0=unfilled,1=low,2=medium,3=high,4=realised}
statusService {0=unfilled,1=low,2=medium,3=high,4=realised}
userReaction {0=none,1=select,2=askMore,3=other}
userSilence={0=false,1=true}
Actions
IP: compare, recommend, summarise, summariseCompare, summariseRecommend, summariseCompareRecommend,
Slot-ordering: presentCuisine, presentFood, presentLocation, presentPrice, presentService,
Incremental: backchannel, correct, selfCorrect, holdFloor, waitMore
Goal State 0, 1, 0 ∨ 4, 0 ∨ 4, 0 ∨ 4, 0 ∨ 4, 0 ∨ 4, 1, 0 ∨ 1

Figure 4: The state and action space of the learning agent. The goal state is reached when all items (that the user may be interested in) have been presented.

Once information has been presented to the user, it is *committed* or *realised*. We again represent realised IUs in the agent's state representation, so that it can monitor its own output. The goal of an MDP is to find an optimal policy π^* according to which the agent receives the maximal possible reward for each visited state. We use the Q-Learning algorithm (Watkins, 1989) to learn an optimal policy according to $\pi^*(s) = \arg\max_{a \in A} Q^*(s, a)$, where Q^* specifies the expected reward for executing action a in state s and then following policy π^*.

5 Experimental Setting

5.1 The State and Action Space

The agent's state space needs to contain all information relevant for choosing an optimal IP strategy and an optimal sequence of incremental actions. Figure 4 shows the state and action space of our learning agent. The states contain information on the incremental and presentation status of the system. The variable 'incrementalStatus' characterises situations in which a particular (incremental) action is triggered. For example, a `holdFloor` is generated when the user has finished speaking, but the system has not yet finished its database lookup. A `correction` is needed when the system has to modify already presented information (because the user changed their preferences) and a `selfCorrection` is needed when previously presented information is modified because the system made a mistake (in recognition or interpretation). The variable 'presStrategy' indicates whether a strategy for IP has been chosen. It is 'filled' when this is the case, and 'unfilled' otherwise. The variables representing the status of the cuisine, food, location, price and service indicate whether the slot is of interest to the user (0 means that the user does not care about it), and what input confidence score is currently associated with its value. Once slots have been presented, they are *realised* and can only be changed through a correction or self-correction.

The variable 'userReaction' shows the user's reaction to an IP episode. The user can select a restaurant, provide more information to further constrain the search or do something else. The 'userSilence' variable indicates whether the user is speaking or not. This can be relevant for holding the floor or generating backchannels. The action set comprises IP actions, actions which enable us to learn the ordering of slots, and actions which allow us to capture incremental phenomena. The complete state-action space size of this agent is roughly 3.2 million. The agent reaches its goal state (defined w.r.t. the state variables in Figure 4) when an IP strategy has been chosen and all relevant attributes have been presented.

5.2 The Simulated Environment

Since a learning agent typically needs several thousand interactions to learn a reasonable policy, we train it in a simulated environment with two components. The first one deals with different IP strategies generally (not just for the incremental case), and the second one focuses on incrementally updated user input hypothesis during the interaction.

To learn a good IP strategy, we use a user simulation by Rieser et al. (2010),[2] which was estimated from human data and uses bi-grams of the form $P(a_{u,t}|IP_{s,t})$, where $a_{u,t}$ is the predicted user reaction at time t to the system's IP strategy $IP_{s,t}$ in state s at time t. We distinguish the user reactions of

also be the responsibility of a dialogue manager.

[2]The simulation data are available from `http://www.classic-project.org/`.

select a restaurant, *addMoreInfo* to the current query to constrain the search, and *other*. The last category is considered an undesired user reaction that the system should learn to avoid. The simulation uses linear smoothing to account for unseen situations. In this way, we can then predict the most likely user reaction to each system action.

While the IP strategies can be used for incremental and non-incremental NLG, the second part of the simulation deals explicitly with the dynamic environment updates during an interaction. We assume that for each restaurant recommendation, the user has the option of filling any or all of the attributes *cuisine, food quality, location, price range* and *service quality*. The possible values of each attribute and possible confidence scores are shown in Table 2 and denote the same as described in Section 5.1.

At the beginning of a learning episode, we assign each attribute a possible value and confidence score with equal probability. For food and service quality, we assume that the user is never interested in bad food or service. Subsequently, confidence scores can change at each time step. (In future work these transition probabilities will be estimated from a data collection, though the following assumptions are realistic, based on our experience.) We assume that a confidence score of 0 changes to any other value with a likelihood of 0.05. A confidence score of 1 changes with a probability of 0.3, a confidence score of 2 with a probability of 0.1 and a confidence score of 3 with a probability of 0.03. Once slots have been realised, their value is set to 4. They cannot be changed then without an explicit correction. We also assume that realised slots change with a probability of 0.1. If they change, we assume that half of the time, the user is the origin of the change (because they changed their mind) and half of the time the system is the origin of the change (because of an ASR or interpretation error). Each time a confidence score is changed, it has a probability of 0.5 to also change its value. The resulting input to the NLG system are data structures of the form *present(cuisine=Indian), confidence=low*.

5.3 The Reward Function

The main trade-off to optimise for IP in an incremental setting is the *timing and order of presentation*. The agent has to decide whether to present information as soon as it becomes available or else wait until confidence for input hypotheses is more stable. Alternatively, it can reorder information to account for different confidence scores. We assign the following rewards[3]: $+100$ if the user selects an item, 0 if the user adds more search constraints, -100 if the user does something else or the system needs to self-correct,-0.5 for holding the floor, and -1 otherwise. In addition, the agent receives an increasing negative reward for the waiting time, $waiting\_time^2$ (to the power of two), in terms of the number of time steps passed since the last item was presented. This reward is theoretically $-\infty$. The agent is thus penalised stronger the longer it delays IP. The rewards for user reactions are assigned at the end of each episode, all other rewards are assigned after each time step. One episode stretches from the moment that a user specifies their initial preferences to the moment in which they choose a restaurant. The agent was trained for 10 thousand episodes.

Attribute	Values	Confidence
Cuisine	Chinese, French, German, In-, dian, Italian, Japanese, Mexican, Scottish, Spanish, Thai	0, 1, 2, 3, 4
Food	bad, adequate, good, very good	0, 1, 2, 3, 4
Location	7 distinct areas of the city	0, 1, 2, 3, 4
Price	cheap, expensive, good-price-for-value, very expensive	0, 1, 2, 3, 4
Service	bad, adequate, good, very good	0, 1, 2, 3, 4

Table 2: User goal slots for restaurant queries with possible values and confidence scores.

6 Experimental Results

After training, the RL agent has learnt the following incremental IP strategy. It will present information slots as soon as they become available if they have a medium or high confidence score. The agent will then order attributes so that those slots with the highest confidence scores are presented first and slots with lower confidence are presented later (by which time they may have achieved higher confidence). If no information is known with medium or high con-

[3]Handcrafted rewards are sufficient for this proof-of-concept study, and can be learned from data for future models (Rieser and Lemon, 2011).

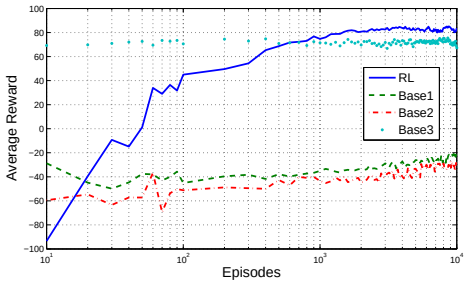

Figure 5: Performance in terms of rewards (averaged over 10 runs) for the RL agent and its baselines.

fidence, the agent will hold the floor or wait. In this way, it can prevent self-corrections and minimise waiting time—both of which yield negative rewards. It can thus start speaking very early (avoiding long pauses or semantically empty utterances) and still has a low likelihood of having to self-correct.

For a comparison of the learnt policy with possible hand-crafted policies (because current incremental NLG systems are rule-based), we designed three baselines. **Baseline 1** always presents information as soon as it is available, i.e. never waits. **Baseline 2** always waits until all information is known with high confidence (i.e. until all confidence scores are 3). **Baseline 3** was chosen to be more ambitious. It always presents information as soon as possible, using a decreasing order of confidence to minimise self-corrections (i.e. very similar to the learnt policy). It chooses randomly among slots with equal confidence. All baseline policies have an optimised IP strategy (recommend/summary etc.) and differ only in their incremental processing strategies. Baseline 1 is most similar to the current approach used in spoken dialogue systems, where the Dialogue Manager triggers an NLG component as soon as a task-relevant user utterance is processed.

Here we do not compare the different strategies for IP generally because this has been done by Rieser et al. (2010), even if not for incremental NLG. Figure 5 shows the performance of all behaviours in terms of average rewards. Baseline 1 obtains on average roughly 100 reward points less than the RL agent. This corresponds to the (negative) reward of one self-correction (-100). Since information is always presented as soon as it is available, this baseline needs to produce on average one

Baseline 1 (present information as soon as it becomes available)
USR Please show me Italian places
($cuisine = Indian; conf\_score = 0.2$)
SYS I have found 35 Indian ...
USR No, ITALIAN
($cuisine = Italian; conf\_score = 0.8$)
SYS Oh, you meant Italian. I have 104 Italian places in Edinburgh. 24 in the city centre, 21 in the New Town, 14 in ...
USR Show me some in the city centre
($location = centre; conf\_score = 0.6$)
SYS OK. I found 24 Italian restaurants in the city centre ...

Baseline 2 (always wait until confidence is high)
USR Do you have Italian restaurants in the centre of town?
$cuisine = Italian; conf\_score = 0.4$
$location = centre; conf\_score = 0.2$
SYS *waits*
USR Italian in the centre.
$cuisine = Italian, conf\_score = 0.7$
$location = centre, conf\_score = 0.5$
SYS I have 104 Italian restaurants.
USR *waits*
SYS *waits*
USR city centre please
$location = centre, conf\_score = 0.7$
SYS I have 24 Italian restaurants in the city centre ...

Baseline 3 (present information in decreasing order of confidence)
USR I want Italian food ...
$cuisine = Indian, conf\_score = 0.2$
$location = centre, conf\_score = 0.3$
SYS hmm (holding turn) ...
USR in the centre of town
$location = centre, conf\_score = 0.9$
SYS In the centre, let me see, Indian ...
USR Italian, please.
$cuisine = Italian, conf\_score = 0.7$
SYS Oh I see. I have 24 Italian places in the centre ...

Figure 6: Example dialogues generated with the baseline policies for a user who wants Italian food in the city centre. Confidence scores for cuisine and location variables for the restaurants are shown as updated.

self-correction per episode. Baseline 2 needs to wait until all information is known with high confidence and obtains on average 125 to 130 rewards less than the RL agent. This corresponds to approximately 11 time steps of waiting (for input to reach higher confidence) before presentation since 11 is (approximately) the square root of 130. Baseline 3 is roughly a reward of -10 worse than the RL agent's be-

haviour, which is due to a combination of more self-corrections, even if they just occur occasionally, and a higher number of turn holding markers. The latter is due to the baseline starting to present as soon as possible, so that whenever all confidence scores are too low to start presenting, a turn holding marker is generated. The learning agent learns to outperform all baselines significantly, by presenting information slots in decreasing order of confidence, combined with waiting and holding the floor at appropriate moments. Anticipating the rewards for waiting vs. holding the floor at particular moments is the main reason that the learnt policy outperforms Baseline 3. Subtle moments of timing as in this case are difficult to hand-craft and more appropriately balanced using optimisation. An absolute comparison of the last 1000 episodes of each behaviour shows that the improvement of the RL agent corresponds to 126.8% over Baseline 1, to 137.7% over Baseline 2 and to 16.76% over Baseline 3. All differences are significant at $p < 0.001$ according to a paired t-test and have a high effect size $r > 0.9$. The high percentage improvement of the learnt policy over Baselines 1 and 2 is mainly due to the high numeric values chosen for the rewards as can be observed from their qualitative behaviour. Thus, if the negative numeric values of, e.g., a self-correction were reduced, the percentage reward would reduce, but the policy would not change qualitatively. Figure 1 shows some examples of the learnt policy including several incremental phenomena. In contrast, Figure 6 shows examples generated with the baselines.

7 Conclusion and Future Directions

We have presented a novel framework combining incremental and statistical approaches to NLG for interactive systems. In a proof-of-concept study in the domain of Information Presentation, we optimised the *timing and order* of IP. The learning agent optimises the trade-off of whether to present information as soon as it becomes available (for high responsiveness) or else to wait until input hypotheses were more stable (to avoid self-corrections). Results in a simulated environment showed that the agent learns to avoid self-corrections and long waiting times, often by presenting information in order of decreasing confidence. It outperforms three hand-crafted baselines due to its enhanced adaptivity. In previous work, incremental responsiveness has mainly been implemented by producing semantically empty fillers such as *um, let me see, well*, etc. (Skantze and Hjalmarsson, 2010). Our work avoids the need for these fillers by content reordering.

Since this paper has focused on a proof-of-concept study, our goal has not been to demonstrate the superiority of automatic optimisation over hand-crafted behaviour. Previous studies have shown the advantages of optimisation (Janarthanam and Lemon, 2010; Rieser et al., 2010; Dethlefs et al., 2011). Rather, our main goal has been to demonstrate that incremental NLG can be phrased as an optimisation problem and that reasonable action policies can be learnt so that an application within an incremental framework is feasible. This observation allows us to take incremental systems, which so far have been restricted to deterministic decision making, one step further in terms of their adaptability and flexibility. To demonstrate the effectiveness of a synergy between RL and incremental NLG on a large scale, we would like to train a fully incremental NLG system from human data using a data-driven reward function. Further, an evaluation with human users will be required to verify the advantages of different policies for Information Presentation.

Regarding the scalability of our optimisation framework, RL systems are known to suffer from the *curse of dimensionality*, the problem that their state space grows exponentially according to the number of variables taken into account. While the application of flat RL is therefore limited to small-scale problems, we can use RL with a divide-and-conquer approach, *hierarchical RL*, which has been shown to scale to large-scale NLG applications (Dethlefs and Cuayáhuitl, 2011), to address complex NLG tasks.

Future work can take several directions. Currently, we learn the agent's behaviour offline, before the interaction, and then execute it statistically. More adaptivity towards individual users and situations could be achieved if the agent was able to learn from ongoing interactions using *online learning*. In addition, current NLG systems tend to assume that the user's goals and situational circumstances are known with certainty. This is often an unrealistic assumption that future work could address using POMDPs (Williams and Young, 2007).

Acknowledgements

The research leading to this work has received funding from EC's FP7 programmes: (FP7/2011-14) under grant agreement no. 287615 (PARLANCE); (FP7/2007-13) under grant agreement no. 216594 (CLASSiC); (FP7/2011-14) under grant agreement no. 270019 (SPACEBOOK); (FP7/2011-16) under grant agreement no. 269427 (STAC).

References

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proc. of EMNLP*, pages 502–512.

Timo Baumann and David Schlangen. 2011. Predicting the Micro-Timing of User Input for an Incremental Spoken Dialogue System that Completes a User's Ongoing Turn. In *Proc. of 12th Annual SIGdial Meeting on Discourse and Dialogue*, Portland, OR.

Timo Baumann, Okko Buss, and David Schlangen. 2011. Evaluation and Optimisation of Incremental Processors. *Dialogue and Discourse*, 2(1).

Anja Belz. 2008. Automatic Generation of Weather Forecast Texts Using Comprehensive Probabilistic Generation-Space Models. *Natural Language Engineering*, 14(4):431–455.

Okko Buss and David Schlangen. 2011. DIUM—An Incremental Dialogue Manager That Can Produce Self-Corrections. In *Proc. of the Workshop on the Semantics and Pragmatics of Dialogue (SemDIAL / Los Angelogue)*, Los Angeles, CA.

Okko Buss, Timo Baumann, and David Schlangen. 2010. Collaborating on Utterances with a Spoken Dialogue Systen Using an ISU-based Approach to Incremental Dialogue Management. In *Proc. of 11th Annual SIGdial Meeting on Discourse and Dialogue*.

Nina Dethlefs and Heriberto Cuayáhuitl. 2011. Combining Hierarchical Reinforcement Learning and Bayesian Networks for Natural Language Generation in Situated Dialogue. In *Proc. of the 13th European Workshop on Natural Language Generation (ENLG)*, Nancy, France.

Nina Dethlefs, Heriberto Cuayáhuitl, and Jette Viethen. 2011. Optimising Natural Language Generation Decision Making for Situated Dialogue. In *Proceedings of the 12th Annual Meeting on Discourse and Dialogue (SIGdial)*, Portland, Oregon, USA.

David DeVault, Kenji Sagae, and David Traum. 2009. Can I finish? Learning when to respond to incremental interpretation result in interactive dialogue. In *Proc. of the 10th Annual SigDial Meeting on Discourse and Dialogue*, Queen Mary University, UK.

Srini Janarthanam and Oliver Lemon. 2010. Learning to Adapt to Unknown Users: Referring Expression Generation in Spoken Dialogue Systems. In *Proc. of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 69–78, July.

Anne Kilger and Wolfgang Finkler. 1995. Incremental generation for real-time applications. Technical report, DFKI Saarbruecken, Germany.

Willem Levelt. 1989. *Speaking: From Intenion to Articulation*. MIT Press.

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 *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1552–1561.

Matthew Purver and Masayuki Otsuka. 2003. Incremental Generation by Incremental Parsing. In *Proceedings of the 6th UK Special-Interesting Group for Computational Linguistics (CLUK) Colloquium*.

Antoine Raux and Maxine Eskenazi. 2009. A Finite-State Turn-Taking Model for Spoken Dialog Systems. In *Proc. of the 10th Conference of the North American Chapter of the Association for Computational Linguistics—Human Language Technologies (NAACL-HLT)*, Boulder, Colorado.

Verena Rieser and Oliver Lemon. 2011. *Reinforcement Learning for Adaptive Dialogue Systems: A Data-driven Methodology for Dialogue Management and Natural Language Generation*. Book Series: Theory and Applications of Natural Language Processing, Springer, Berlin/Heidelberg.

Verena Rieser, Oliver Lemon, and Xingkun Liu. 2010. Optimising Information Presentation for Spoken Dialogue Systems. In *Proc. of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, Uppsala, Sweden.

David Schlangen and Gabriel Skantze. 2009. A General, Abstract Model of Incremental Dialogue Processing. In *Proc. of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, Athens, Greece.

Gabriel Skantze and Anna Hjalmarsson. 2010. Towards Incremental Speech Generation in Dialogue Systems. In *Proc. of the 11th Annual SigDial Meeting on Discourse and Dialogue*, Tokyo, Japan.

Gabriel Skantze and David Schlangen. 2009. Incremental Dialogue Processing in a Micro-Domain. In *Proc. of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, Athens, Greece.

Amanda Stent, Rashmi Prasad, and Marilyn Walker. 2004. Trainable sentence planning for complex information presentation in spoken dialogue systems. In

Proc. of the Annual Meeting of the Association for Computational Linguistics.

Kees van Deemter. 2009. What game theory can do for NLG: the case of vague language. In *12th European Workshop on Natural Language Generation (ENLG)*.

Marilyn Walker, Steve Whittaker, Amanda Stent, Pretaam Maloor, Johanna Moore, and G Vasireddy. 2004. Generation and Evaluation of User Tailored Responses in Multimodal Dialogue. *Cognitive Science*, 28(5):811–840.

Chris Watkins. 1989. *Learning from Delayed Rewards*. PhD Thesis, King's College, Cambridge, UK.

Jason Williams and Steve Young. 2007. Partially Observable Markov Decision Processes for Spoken Dialog Systems. *Computer Speech and Language*, 21(2):393–422.