The Bremen System for the GIVE-2.5 Challenge

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

Abstract

This paper presents the Bremen system for the GIVE-2.5 challenge. It is based on decision trees learnt from new annotations of the GIVE corpus augmented with manually specified rules. Surface realisation is based on context-free grammars. The paper will address advantages and shortcomings of the approach and discuss how the present system can serve as a baseline for a future evaluation with an improved version using hierarchical reinforcement learning with graphical models.

1 Introduction

Decision making in NLG systems for situated domains needs to be sensitive to a number of features concerning the spatial context, the user and the history of the interaction. Related work to situated NLG has explored different approaches to this problem. Stoia et al. (2006) use decision trees to learn a set of rules for referring expression generation (REG) in a virtual environment very similar to GIVE (Byron, 2005). Similarly, Dale and Viethen (2008) and Viethen (2010) use decision trees to inform REG in a spatial setting. Garoufi and Koller (2010) use AI planning for GIVE to principally guide the user to positions where unambiguous referring expressions (RE) can be generated. Denis (2010) uses an algorithm based on Reference Domain Theory to generate REs for GIVE based on context. Finally, Benotti and Denis (2011) use a corpus-based selection method to choose utterances from a human corpus to present to the user. In Dethlefs et al. (2011), we suggested to use Hierarchical Reinforcement Learning (HRL) for GIVE and compared it against decision trees. While results (based on simulation and human ratings) showed that the HRL system achieved significantly better performance, this paper presents a system that behaves based on decision trees learnt from human data. The system is developed as a reliable baseline for a comprehensive evaluation of an HRL-based system in the future (as part of the author's PhD thesis).

2 The GIVE Task

The GIVE task involves the generation of navigation instructions and REs in a virtual 3D world (Koller et al., 2010), where two participants go on a 'treasure hunt'. One participant instructs the other in navigating through the world, pressing a sequence of buttons and completing the task by obtaining a trophy.

2.1 GIVE-2 Corpus Annotation

While typically the task of instruction giver is taken by an NLG system, the GIVE-2 corpus (Gargett et al., 2010) provides 63 English and 45 German transcripts of human-human dialogues for the task. To design an NLG system for GIVE and automatically induce a set of rules to inform its design, the English dialogues were complemented with a set of semantic annotations. They include the string of words and time of an utterance as well as its type. Utterance types include *destination, direction, orientation, path* and *'straight'* for navigation and *manipulation, confirm* and *stop* otherwise. High-level navigation (e.g., 'go back to the previous room') and low-level navigation (e.g., 'go straight, then

Proceedings of the 13th European Workshop on Natural Language Generation (ENLG), pages 284–289,
Nancy, France, September 2011.

Utterance
string=“turn left and press the blue button left of the yellow”, *time*=‘20:54:55’

Utterance_type
content=‘orientation,manipulation’ [straight, path, direction, destination, confirm, stop]
navigation_level=‘low’ [high]

Referring_Expression
first_mention=‘true’ [false], *within_field_of_vision*=‘true’ [false]
discriminative_colour_distractor=‘true’ [false], *mention_distractor_colour*=‘true’ [false]
discriminative_colour_referent=‘false’ [true], *mention_referent_colour*=‘true’ [false]
mention_distractor=‘true’ [false] , *mention_landmark*=‘false’ [true]
spatial_relation=‘lateral_projection’ [none, distance, middle, proximal, functional_control, functional_containment, non_projection_axial, frontal_projection, vertical_projection]

User
user_position=‘on_track’ [off_track],
user_reaction=‘perform_desired_action’ [perform_undesired_action, wait, request_help]

Figure 1: Sample annotation for a navigation instruction followed by a referring expression. Alternative annotation values are given in square brackets behind the actual values.

left and turn right’) is distinguished. The former refers to contractions of the later. In terms of referring expressions (or *manipulation* utterances), annotations include whether a referent has been mentioned before, whether it has a discriminating colour, whether it has a distractor with a discriminating colour, whether a distractor or landmark was included in an utterance, whether the referent is visible and the type of spatial relation between a distractor or landmark and the referent. Spatial relations were annotated according to Bateman et al. (2010). Please see Figure 1 for an example annotation.

2.2 Generation Tasks

The NLG system was designed to perform four main tasks. **(1) High-level behaviour generation** is concerned with deciding what type of utterance to generate next among navigation instructions, referring expressions, confirmations or stop instructions. **(2) Navigation instruction generation** chooses a level of navigation (high or low), according to the degree of confusion of the user and their prior knowledge of the virtual world. **(3) REG** includes deciding to mention a referent’s colour or not, mention a distractor (and its colour) or not, mention a landmark or not, and deciding what spatial relation to use (if any). **(4) Surface Realisation** produces a string of words for presentation to the user from the semantics determined by the previous components.

3 Algorithms for Content Selection

3.1 Learning Decision Trees

To learn a set of rules from the annotated GIVE corpus, Weka’s (Witten and Frank, 2005) J48 classifier was used. We learnt one decision tree per annotated attribute. Rules for navigation instructions were learnt based on utterance type and user features, and RE rules were learnt based on the RE and user features. On average, the decision trees reached an accuracy of 91% in a 10-fold cross validation. The obtained rules were integrated into the algorithms designed for each behaviour.

3.2 High-level Behaviour

The high-level behaviour of the system was entirely hand-crafted. Whenever a game is started, the system greets the user and introduces them to the main task of the game. A first warning is then presented to the user to not step on any red tiles. After this first warning, additional warnings are generated whenever an alarm tile is visible and near to the user (so there is eminent danger of activating an alarm) or when a tile is visible and less than five warnings have been generated during the whole interaction. The objective of generating multiple warnings was to raise a strong awareness of their danger. Moreover, the system confirms successful manipulation actions of the user (to convey a notion of progress in the game), but not for successful navigation instruc-

Algorithm 1 Algorithm for generating navigation instructions.

```
1: function GENERATENAVIGATION(userConfusions c, nextGoal g, boolean leaving_room) return navigation
2:     instruction_type ← instruction type of destination, path, direction, orientation and straight
3:     navigation_level ← instruction level of high and low
4:     while navigation is not generated do
5:         if next room is known and userConfusions c = 0 then
6:             navigation_level = high
7:         else
8:             navigation_level = low
9:         end if
10:        if user is leaving_room is true and number of doors >1 then
11:            instruction_type = path + direction
12:        else if user is leaving_room is true and number of doors = 1 then
13:            instruction_type = path
14:        else if the user is leaving_room is true and the nextGoal g is in the same room then
15:            instruction_type = destination towards object
16:            if a salient landmark is present near the next goal then
17:                object = landmark
18:            else if a door is present then
19:                object = door
20:            else
21:                object = nextGoal g
22:            end if
23:        else if the user is leaving the room over a corridor then
24:            instruction_type = path
25:        else if the user is changing their orientation then
26:            instruction_type = orientation
27:        else if the user is going straight then
28:            instruction_type = straight
29:        else if the user is heading to another direction then
30:            instruction_type = direction
31:        end if
32:    end while
33: end function
```

tions (to not interrupt smooth interactions). Whenever the user requests help (by pressing the help button), the system either repeats the previous utterance or generates a paraphrase. The same behaviour is shown for user confusions (which we assume after five seconds that users do not do anything).

3.3 Navigation Instructions

Navigation instruction generation is partially learnt from decision trees and partially hand-crafted. It specifies that the agent should try to use high-level navigation behaviour whenever this is likely to be successful (i.e. when the user is not confused and the next room is already known). High-level instructions in this case could encourage shorter and more efficient interactions. Whenever the user leaves a room and a door needs to be mentioned, the direction of the door is included when there is more than one in the room. If a destination instruction to some object is given, the system prefers instructions to salient landmarks of the environment over buttons (since landmarks tend to be less ambiguous). Whenever no landmarks are present and a destination instruction to a button is generated, using the next referent as a destination is preferred over using a distractor. The resulting algorithm is shown in Algorithm 2. The behaviour specified in lines 5-9 (on high-level navigation) and in lines 16-22 (about choosing a salient object) were hand-crafted, the remaining behaviour was learnt.

Algorithm 2 Algorithm for generating referring expressions.

1: **function** GENERATERE(referent r, distractors $d_{0...n}$, landmarks $l_{0...m}$) **return** RE
2: int $reminders \leftarrow$ reminders to the user of getting close to r when pressing
3: **while** RE is not generated **do**
4: **if** r is visible and near **then**
5: **if** utterance is of type $repair$ and $colour$ of r is discriminating **then**
6: include $colour$ of r
7: **else if** utterance is not of type $repair$ **then**
8: include $colour$ of r
9: **else**
10: don't include $colour$ of r
11: **end if**
12: **if** $colour$ of r is not discriminating and number of distractors $d_{0...n}$ is not 0 **then**
13: **for** d_i in $d_{0...n}$ **do**
14: **if** $colour$ of distractor d_i is discriminating and d_i is adjacent to r **then**
15: include d_i and $colour$ of d_i
16: **else if** $spatial\_relation$ between d_i and r is vertical **then**
17: include d_i but not $colour$ of d_i
18: **else if** $spatial\_relation$ between d_i and r is lateral or horizontal **then**
19: include d_i but not $colour$ of d_i
20: **end if**
21: **end for**
22: **else if** $colour$ of r is not discriminating and number of landmarks $l_{0...m}$ is not 0 **then**
23: include l_j that is closest to r
24: **end if**
25: **else if** d_i is visible and near but r is not **then**
26: RE = 'Not this one, I mean the other button.'
27: **else**
28: RE = 'Try to find a button somewhere near.'
29: **end if**
30: **if** reminders <5 **then**
31: RE = RE + 'Remember to get really close to press it.'
32: **end if**
33: **end while**
34: **end function**

3.4 Referring Expressions

The REG behaviour is again partially learnt and partially hand-crafted. The system mentions the colour of a referent whenever the current utterance type is not a repair or if the referent's colour is discriminating. If it is not, the system's next best choice is to use a distractor with a discriminating colour that is adjacent to the referent. Otherwise, it prefers to locate the referent using a vertical spatial relation over using a lateral or horizontal one. If no suitable distractor is present, a referent can be located with respect to a landmark. This set of rules was entirely learnt from decision trees (lines 6-25 in Algorithm 3). The remaining behaviour was designed manually. Whenever a distractor is near to the user and the only button visible, it was assumed that the user had the intention of pressing it. A warning is generated in this case that this was the wrong button. If no button is visible or near, the user is told to look for one in the vicinity. In addition, the system initially generates a set of reminders (up to five) to the user to get close enough to a button before pressing.

4 CFGs for Surface Realisation

The generation spaces of the system were represented as CFGs so that several alternative realisations of a semantic concept could be captured and alternated for more variable system output. In order to

CFG Generation Space for destination instructions

destination1 → desVerb1 desPrep1 desRel1
destination2 → desVerb1 desPrep2 desRel2
destination3 → desVerb2 desRel1|desRel2
desVerb1 → go | keep_going | walk | continue | empty
desVerb2 → you_need | you_want | get
desPrep1 → to | towards | until
desPrep2 → into | in
desRel1 → pointRelatum
desRel2 → roomRelatum

Figure 2: Example CFG for destination instruction. Non-terminal symbols represent semantic constituents, terminal symbols possible surface realisations.

obtain CFGs, we used the ABL algorithm (van Zaanen, 2000), which aligns strings based on Minimum Edit Distance and induces a CFG automatically from the aligned examples. The annotated GIVE corpus examples were used as input to the algorithm based on their instruction type, so that separate generation spaces were obtained for destination, direction, orientation, path and 'straight' instructions as well as REs. As an example, the CFG for destination instructions is shown in Figure 2. Here, a destination instruction can be phrased in three different ways. Type 1 generates instructions such as 'Go to the sofa' (referring to point-like destinations) and type 2 generates instructions such as 'Go into the next room' (referring to room-like destinations). Type 3 destination instructions use verb forms which are followed directly by either type of relatum. We use these CFGs to generate variation in surface forms.[1]

5 Results

The GIVE-2.5 evaluation revealed advantages as well as drawbacks of the presented approach. Some of the drawbacks that users commented on involved ambiguities with respect to doors or button referents, which were not identified uniquely, or the disambiguation occurred too late (e.g. when the system first generated an ambiguous phrase and then repaired it with an unambiguous paraphrase). This aspect would usually not affect task success measures, but can deteriorate user satisfaction scores. In terms of task success, the system reached roughly 56% which is a better number than any system achieved in the 2010 challenge, but is a bad number in comparison to the 2011 systems. This low number provides strong evidence that the generated tile warnings were insufficient, since a number of users lost the game here. Both points of criticism can be traced back to a lack in flexibility in system behaviour. A system with better (more adaptive) troubleshooting strategies to lead the user around tiles (instead of warning them) and avoiding ambiguous phrases at any time would likely have reached higher task success scores. This also affects positive feedback that users provided on the high-level navigation strategies. Several users stated that they wished the system had employed this strategy more globally.

6 Discussion and Conclusion

The reason for the system's lack of troubleshooting strategies and limited flexibility is likely found in the method itself: the human corpus data from which the decision trees were learned presented data sparsity problems with respect to learning troubleshooting strategies. This is because human users in the corpus react very flexibly to individual problematic situations which may stretch over several turns and therefore not be captured by the annotations.[2]

A powerful alternative to decision trees is reinforcement learning (or HRL for large systems) which has been applied to situated interaction (Cuayáhuitl and Dethlefs, 2011; Dethlefs et al., 2011) with promising results. Since RL agents are able to learn flexible behaviour strategies from a limited amount of data (using simulations), they often do not face the same data sparsity problems as supervised learning accounts. Rule-based systems may present a viable alternative for small and limited domains, but will not scale to complex real-world problems because of the large amount of manual work they require. Corpus-based selection methods that have been proposed recently (Benotti and Denis, 2011) appear to yield good results for clearly pre-specified tasks but will not present an alternative for tasks involving uncertainty or the need to gener-

[1] This variation is random in that it is not based on the likelihoods with which different forms appear in the human data.

[2] A more comprehensive annotation scheme could possibly improve performance in this case, but would probably not solve the data sparsity problems on the whole.

alise to new circumstances.

7 Future Directions

The presented system suffered from a number of drawbacks that future work will address. We observed a lack of flexibility in the system's behaviour especially when sophisticated troubleshooting strategies were needed. A hypothesis is that a system based on hierarchical RL could adapt more flexibly to different (unseen) conditions and provide better support to individual users. In addition, graphical models such as Bayesian Networks (Dethlefs and Cuayáhuitl, 2011a) or HMMs (Dethlefs and Cuayáhuitl, 2011b) can be used to formulate more sophisticated generation spaces based on corpus probabilities and support more coherent surface realisation. Both claims will be tested and evaluated against the baseline established in this paper.

Acknowledgments

Thanks to the German Research Foundation DFG and the Transregional Collaborative Research Centre SFB/TR8 'Spatial Cognition' for partial support.

References

John A. Bateman, Joana Hois, Robert Ross, and Thora Tenbrink. 2010. A linguistic ontology of space for natural language processing. *Artificial Intelligence*, 174(14):1027 – 1071.

Luciana Benotti and Alexandre Denis. 2011. Giving instructions in virtual environments by corpus based selection. In *Proceedings of the 12th Annual SIGdial Meeting on Discourse and Dialogue*.

Donna Byron. 2005. The osu quake 2004 corpus of two-party situated problem-solving dialogs. In *Technical Report OSU-CISRC-805-TR57, The Ohio State University Computer Science and Engineering Department*.

Heriberto Cuayáhuitl and Nina Dethlefs. 2011. 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).

Robert Dale and Jette Viethen. 2009. Referring expression generation through attribute-based heuristics. In *Proceedings of the 12th European Workshop on Natural Language Generation*, ENLG '09, pages 58–65.

Alexandre Denis. 2010. Generating referring expressions with reference domain theory. In *Proceeding of the 6th International Conference on Natural Language Generation (INLG)*.

Nina Dethlefs and Heriberto Cuayáhuitl. 2011a. Combining hierarchical reinforcement learning and bayesian networks for situated dialogue. In *Proceedings of the 13th European Workshop on Natural Language Generation (ENLG), Nancy, France*.

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

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

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)*.

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

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.

Laura Stoia, Darla Magdalene Shockley, Donna K. Byron, and Eric Fosler-Lussier. 2006. Noun phrase generation for situated dialogs. In *Proceedings of the Fourth International Natural Language Generation Conference*, INLG '06, pages 81–88, Morristown, NJ, USA.

Menno van Zaanen. 2000. Bootstrapping syntax and recursion using alginment-based learning. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, pages 1063–1070, San Francisco, CA, USA.

Jette Viethen. 2010. *Generating Natural Descriptions: Corpus-Based Referring Expression Generation in Visual Domains*. Ph.D. thesis, Macquarie University, Sydney, Australia.

Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, CA, 2. edition.