The Thumbs Up! Twente system for GIVE 2.5

Saskia Akkersdijk, Marin Langenbach, Frieder Loch, Mariët Theune
Human Media Interaction
University of Twente
P.O. Box 217, 7500 AE, Enschede, The Netherlands
{s.m.akkersdijk|m.langenbach|f.loch}@student.utwente.nl, m.theune@utwente.nl

Abstract

This paper describes the Thumbs Up! Twente system, a natural language generation system designed for the GIVE 2.5 Challenge. The purpose of the system is to guide a user through a virtual 3D environment by generating instructions in real-time. Our system focuses on motivating the user to keep him playing the game and trying to find the trophy.

1 Introduction

This report describes a natural language generation system called **Thumbs Up! Twente** (TU!T). It was developed for the Generating Instructions in Virtual Environments (GIVE) 2.5 Challenge,[1] which involves generating instructions that guide users to press coloured buttons and walk around the different rooms of a 3D-world. The goal is to find a trophy, which is located in a safe that can be opened by pressing a particular sequence of buttons.

Our system focuses on motivating the users through feedback to keep them playing. Before addressing this, we first describe other important aspects such as planning and the generation of instructions and referring expressions. We end with a presentation and discussion of evaluation results.

2 Planning

The basis for instruction generation in GIVE is a plan: a sequence of actions, created by a planner that was provided by the GIVE organisation. Each room in the 3D-world is divided into regions, and the initial plan consists of separate move actions for each region; see Figure 1 (left). TU!T aggregates these separate steps to enable the generation of high-level navigation instructions. This has several advantages (Braunias et al., 2010; McCoy et al., 2010):

- The users are free to choose their own way towards the instruction target, making the task more interesting.
- Fewer instructions are needed, leaving the user more time to read and understand the instructions, and (in the case of TU!T) leaving more room for motivational feedback.

Steps between regions in the same room are aggregated by TU!T as shown in Figure 1 (right). During aggregation it is checked whether the target region is still visible from the user's position. If this is not the case, plan steps are aggregated until the last visible region (see Figure 2).

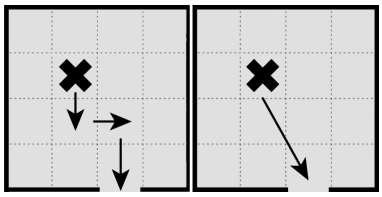

Figure 1: Aggregation of plan steps based on rooms.

3 Instruction Generation

When turning plan steps into verbal instructions, we tried to keep the instructions short and simply structured, because overly long instructions turned out

[1] http://www.give-challenge.org/research

Proceedings of the 13th European Workshop on Natural Language Generation (ENLG), pages 312–317,
Nancy, France, September 2011.

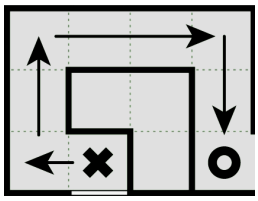

Figure 2: Aggregation of plan steps based on visibility of the target region.

to be a source of problems for systems from earlier challenges. According to the GIVE 2 report (Koller et al., 2010), the systems with the highest task success rate were those that produced the shortest instructions. We do not vary the wording of the instructions, because this might increase the difficulty in following and reading them quickly.

TU!T distinguishes three main instruction types (besides the final instruction to take the trophy):

- **Instructions to press a button** consist of the word *Press* followed by a reference to the target button, as described in Section 4.1.
- **Instructions to move to another room** refer to the door that the user needs to move through, as described in Section 4.2. This is the kind of situation shown in Figure 1.
- **Instructions to move to a location in the current room** are given in the kind of situation from Figure 2. They start with *Move around the corner,* since in most (but not all) cases the target region is, indeed, located around a corner. The direction in which the target region is located is added to form instructions such as *Move around the corner to your left.*

Note that TU!T also generates references to doors and buttons that are not currently visible to the user. If the target is behind the user, an instruction is added on how the user should turn to see it, e.g., *Press the blue button behind you. Turn around and go left.* If the instruction length does not exceed a certain threshold, explicit information about the visibility of the target is included in the instruction.

At fixed intervals, and after each button press, the system checks whether the user is still following the plan. If so, the generation of the next instruction is triggered. If the user has moved to the wrong room, TU!T informs the user of this. The user is given 6 seconds to move to the correct room without new instructions. If after this 'patience period' (Schütte and Dethlefs, 2010) the user has not corrected the mistake, a new plan is created starting from the user's current location. This is also done after the user has pressed a wrong button.

TU!T incorporates a mechanism to generate a new version of the current instruction when the user moves closer to the target. This may be helpful because at a shorter distance, referring expressions tend to become more specific. If the user is still at some distance the instruction may be fairly general, for instance *Press the blue button to your left* allowing the user to globally locate the button. When the instruction is repeated at a shorter distance it will generate a unique description, for instance *Press the left button in the middle row*, making it possible to successfully identify the target button. An updated version of the current instruction is also generated when the user presses "h" to call the *Help* function.

4 Referring Expression Generation

Referring expression generation (REG) in the GIVE worlds mainly involves referring to buttons and doors. We considered using the graph-based algorithm for this (Krahmer et al., 2003), but it turned out to be too slow for real-time, on-the-fly generation of expressions as required in GIVE. So we created our own referring expression generator, incorporating lessons learned from the GIVE 2 systems.

4.1 Referring to Buttons

TU!T uses two different methods for referring to buttons, *SimpleREG* and *GridREG*. Both methods always include the button's colour in the description, even when not strictly necessary for identification. This is done to prevent any possible confusion, and because humans tend to mention redundant properties as well (Dale and Reiter, 1995).

SimpleREG looks for landmarks around the button. We never include more than one landmark, otherwise the expressions might get too long and potentially confusing (Schütte and Dethlefs, 2010). Landmarks are selected by searching for the object closest to the target object, but some candidates are

discarded. Open doors are never used, and buttons are only used as landmarks if there cannot be any confusion as to which button is meant. See Figure 3 for an example, where the lamp was chosen as the best landmark. We also include information about the button's location relative to the user, to form instructions such as *Press the green button on your right and to the right of the lamp.*

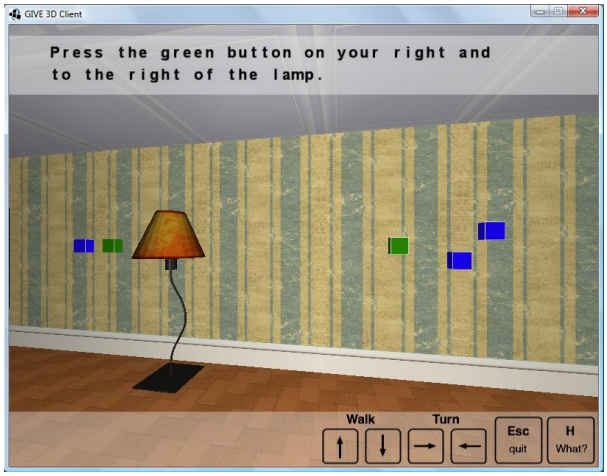

Figure 3: Referring to a button using a landmark.

GridREG creates a grid from all buttons, and counts them from left to right and from top to bottom. If there are, for example, nine buttons in a 3x3 grid the algorithm generates instructions such as *Press the blue button, it is the top left one*. If there is only one row of buttons, it generates references such as *Press the blue button, it is the second one from the left*. This approach is similar to that of Braunias et al. (2010) for GIVE 2.

The criterion for using SimpleREG or GridREG is the number of visible buttons with the same colour as the target button. As the name suggests, SimpleREG is used in relatively simple cases, when the target button is visible and maximally one other button of the same colour. SimpleREG is also used if the target button is not visible. In that case, TU!T generates instructions such as *Press the yellow button on your left. You cannot see it.* If there is a visible landmark, it is added to the description. GridREG is used when one or more buttons are visible with the same colour as the target button.

If there is only one other visible button with the same colour as the target button (as in Figure 3), one of the two methods is randomly selected, because they are equally suitable. TU!T records which method was used for the initial description, so that if the user presses "h", the other method can be used. This way the *Help* function can really clarify the situation if the user is confused, instead of only repeating the exact same instruction.

4.2 Referring to Doors

Referring to doors in the world is relatively simple, because their only distinguishing property is their location. TU!T never uses landmarks in connection to doors. If only the target door is visible, TU!T simply always says *Move through the door*. If the target door is not visible, its position relative to the user is mentioned, leading to instructions such as *Move through the door behind you*. If more than one door is visible, GridREG is used in a similar way as for buttons. But for door references an extra feature was added: TU!T searches for a hallway by looking at the coordinates of all visible doors in the user's current room, and checking if they are aligned in two rows. In this case the doors on each side are counted, to create instructions such as *Move through the second door on your right*.

We only include doors in the same room as the target door in the context set. If more doors are visible through an open door, these are not taken into account. This should be less confusing for the users, who probably assume they need to use a door inside the current room. We only consider open doors, because users never have to go through closed doors.

5 Motivation through Feedback

Keeping the user motivated is one of the main goals of our system. A motivated user is less likely to give up, which should reduce the number of canceled games and increase the number of successful games. Also, the overall experience of the user will be more positive. The way we make our system motivating is by giving two types of feedback.

Reflective feedback reports on the user's progress, triggered by a timer. The system randomly chooses a fitting feedback sentence, based on the

number of remaining buttons to be pressed.[2] Examples are *You are in the second half of this game* and *Almost there!* The second type of reflective feedback is positive feedback after a correct action, which is known to enhance motivation (Harackiewicz, 1979; Vallerand and Reid, 1988). Examples are sentences such as *Well done!* and *Good job!* Finally, reflective feedback is given when a user enters the wrong room, for example *That's not the correct way*.

Anticipating feedback is feedback on what is visible for the user, based on what we think the user wants or needs to know. Confirmation that the user is looking at the correct object can be really useful and can make the user more confident. Telling the user that he/she is looking at the wrong object prevents wrong button presses, and makes navigation through the world more efficient.

When giving anticipating feedback on visible buttons we distinguish five situations:

- **Only the target button is visible:** in this case the system confirms that this is the correct button, for example by saying *Yes, that one.*
- **Only buttons of the wrong colour are visible:** in this case the system reminds the user that he/she needs a button with another colour, for example by saying *No, not this button. It should be blue* (Figure 4A).
- **Only wrong buttons, but of the correct colour are visible:** here, the system tells the user that another button is needed, for example by saying *This is the wrong button* (Figure 4B).
- **The target button is visible, as well as one or more other buttons, all of the wrong colour:** here, the system points out the target button, for example by saying *The blue one is the correct button* (Figure 4C).
- **The target button is visible, as well as one or more other buttons of the same colour:** in this case we give no feedback because it might be confusing (Figure 4D). Also, as the user comes closer, button visibility changes and one of the other situations will apply.

[2]Unlike instruction messages, feedback messages have variants with different wording.

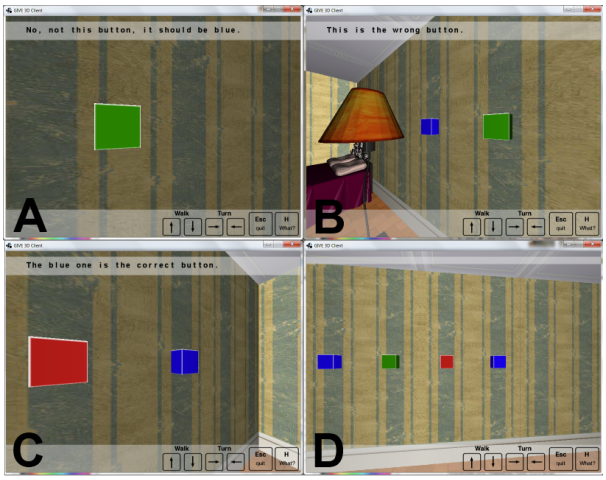

Figure 4: Four anticipating feedback situations.

For anticipating feedback on visible doors we distinguish three situations.

- **Only the target door is visible:** the system gives feedback that it is the correct door, for example by saying *Yes, that doorway*.
- **Another door than the target is visible:** the system tells the user that this is not the correct door, for example by saying *This is the wrong doorway*.
- **The target door and one or more other doors are visible:** in this case we give no feedback, for the same reasons as with buttons.

In addition to feedback on buttons and doors, TU!T also issues warnings when the user approaches an alarm tile. In GIVE 2, the lack of such warnings was identified as a major source of problems by McCoy et al. (2010) and Roth et al. (2010). To prevent irritation, TU!T keeps intervals of at least six seconds between warnings.

6 MessageQueue

The message queue sorts the messages to be displayed in order of importance. For example, navigational messages are always important, while progress feedback is less important. It also keeps track of how long a message should be displayed, which depends on the length of the message. As

long as the queue is not empty, the first (most important) message is taken from the queue and displayed for the given duration.

It can be that while a message is displayed, a more important message is created. Then the current message is stopped and overwritten by the new message, to ensure that the displayed message accurately reflects the current situation. For example, if the user moves toward the target button while the system is giving feedback on a wrong button that was in view, the old feedback is replaced by a new message.

After each button press the message queue is emptied, to prevent it from becoming too full with old messages that may be no longer applicable.

7 Evaluation

GIVE 2.5 used three evaluation worlds, of which World 1 was the simplest. In World 2, the buttons were positioned in grids of different shapes, and World 3 had a large space with many doors, posing a challenge for direction giving. Table 1 shows the TU!T results for the three worlds, based on 22 games in World 1, 16 games in World 2 and 9 games in World 3, played between 1 July - 22 August 2011 in the online GIVE evaluation experiment. The subjective ratings indicate the level of agreement with statements such as "The system's instructions were visible long enough for me to read them" and "The system immediately offered help when I was in trouble." For readability, we reversed the polarity of ratings for negative statements.

TU!T performed relatively well in World 1, but it had problems with the button descriptions in World 2, and its performance in World 3 was bad overall. The evaluation participants did find the system friendly and appreciated its feedback, in particular in Worlds 1 and 2.

Measure	**World 1**	**World 2**	**World 3**
Successful games	68.2%	56.3%	22,2%
Lost games	13.6%	6.0%	44.4%
Cancelled games	18.2%	37.5%	33.3%
Q1: Overall quality	20.3	-6.0	-37.5
Q2: Directions	0.9	-31.9	-24.2
Q3: Button description	45.1	-17.9	19.9
Q4: Instruction clarity	9.4	-1.9	-3.4
Q5: Display duration	21.9	31.3	20.4
Q6: Instruction timing	10.4	3.4	-13.6
Q7: Help immediacy	18.6	-4.1	-27.8
Q8: Feedback	36.1	44.2	1.3
Q9: Friendliness	41.3	29.8	11.7
Q10: Trustworthiness	43.5	-2.4	-49.3

Table 1: Results for the GIVE 2.5 evaluation worlds. Subjective ratings are on a scale of -100 to 100.

8 Discussion

The evaluation results point to various flaws in the system. When referring to doors, TU!T naively assumes that they are aligned on one or two axes (walls). In a room with doors on three or more walls, as in World 3, this leads to confusing expressions. When the user approaches the doors the system's feedback will allow the user to find the correct one eventually, but this is far from efficient. Several evaluation participants commented that they reverted to 'trial and error' navigation when instructions were unclear, relying on the feedback to find out whether they were facing the right door or button.

Currently, TU!T generates non-unique descriptions such as *the blue button in front of you*, which are clarified automatically when the user approaches the target. This first description bears the danger of confusing the user (confirmed by participants' comments). Instead it would be better to first generate a move instruction that guides the user to a position from which a unique referring expression to the target button can be generated. A similar approach could be used for referring to doors.

As noted by Denis et al. (2010), instructions that are relative to the user's position (e.g., *behind you*) can be problematic because users can move through the world quite fast. We tried to make TU!T as fast as possible, but it still suffers from this problem. Sometimes the user moves around quickly and then receives an outdated instruction. The user can press *Help* for a new instruction, but this is an extra action that could be rendered unnecessary by improving the performance of our system.

The feedback mechanism incorporated in TU!T proved to be successful: almost all participants commented positively on this feature, in particular the anticipating feedback. It is hard to verify whether the feedback really had a motivational effect, but it was clearly perceived as helpful.

References

Johannes Braunias, Uwe Boltz, Markus Dräger, Boris Fersing, and Olga Nikitina. 2010. The GIVE-2 Challenge: Saarland NLG System. In *Online Proceedings of the GIVE-2 Challenge*.

Robert Dale and Ehud Reiter. 1995. Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science*, 19(2):233–263.

Alexandre Denis, Marilisa Amoia, Luciana Benotti, Laura Perez-Beltrachini, Claire Gardent, and Tarik Osswald. 2010. The GIVE-2 Nancy Generation Systems NA and NM. In *Online Proceedings of the GIVE-2 Challenge*.

Judith M. Harackiewicz. 1979. The effects of reward contingency and performance feedback on intrinsic motivation. *Journal of Personality and Social Psychology*, 37(8):1352–1363.

Alexander Koller, Kristina Striegnitz, Andrew Gargett, Donna Byron, Justine Cassell, Robert Dale, Johanna Moore, and Jon Oberlander. 2010. Report on the second NLG challenge on Generating Instructions in Virtual Environments (GIVE-2). In *Proceedings of the 6th International Natural Language Generation Conference (INLG 2010)*, pages 243–250.

Emiel Krahmer, Sebastiaan van Erk, and André Verleg. 2003. Graph-based generation of referring expressions. *Computational Linguistics*, 29(1):53–72.

Dermot Hayes McCoy, Ielka van der Sluis, and Saturnino Luz. 2010. The TCD system for GIVE-2. In *Online Proceedings of the GIVE-2 Challenge*.

Michael Roth, Michael Haas, Eric Hildebrand, and Eleftherios Matios. 2010. The Heidelberg GIVE-2 System. In *Online Proceedings of the GIVE-2 Challenge*.

Niels Schütte and Nina Dethlefs. 2010. The Dublin-Bremen System for the GIVE-2 Challenge. In *Online Proceedings of the GIVE-2 Challenge*.

Robert J. Vallerand and Greg Reid. 1988. On the relative effects of positive and negative verbal feedback on males' and females' intrinsic motivation. *Canadian Journal of Behavioural Science/Revue Canadienne des Sciences du Comportement*, 20(3):239–250.