Generating Instructions in Virtual Environments as a Planning Problem

Yehuda Katz, Phil Nguyen, Peratham Wiriyathammabhum

March 11, 2015

Abstract

This project aims to create a Natural Language Generation system for giving instructions in virtual world with different level of abstractions. We will try to incorporate hierarchical planner into the Potsdam Natural Language Generation systems which originally use forward search for planning.

1 Introduction

The Challenge on Generating Instruction in Virtual Environment (GIVE) [7] is an evaluation shared task for real-time natural language generation (NLG) systems. The system objective is to guide a human instruction follower (IF) to complete a treasure-hunting task in a 3D virtual world. We put our interests into the recent version of GIVE, namely, GIVE-2.5 [9]. GIVE-2.5 is the second second sequel of GIVE challenge which is an identical challenge to GIVE-2 [8] but was hold again for a better timing.

Users will start the 3D game from the website. First, users will download the 3D client which allows them to interact with the 3D world. Next, the client will connect to the NLG system. There are 3 evaluation worlds with different difficulties as depicted in Figure. 1. World 1 has a simple layout and buttons are easily identified. World 2 buttons are cluttered with a lot of buttons with the same colors but the user can still refer to the room by the color of the furnitures. World 3 has a maze-like layout in one room, multiple alarm tiles in another room and several doors with a lot of plants in the last room.

The task is for a user to pick up a trophy from a safe which should be preceded by pushing a sequence of buttons. There are some obstacles such as some floor tiles that can generate an alarm signal which will cause a user to lose the game if the alarms are not deactivated in advanced. In addition, there are many unrelated objects such as lamps or plants which are dummy and will be used as landmarks by the NLG systems. The demo can be found at `https://www.youtube.com/watch?v=0RSin0HChno`.

The main different between GIVE and its sequels are the ability for a user to move. In GIVE, there are discrete steps so 'three steps' and 'turn left' will be precise actions. While in GIVE-2 and GIVE-2.5, a user can move freely so the phrases like 'three steps' and 'turn left' will be more difficult to predict their effects.

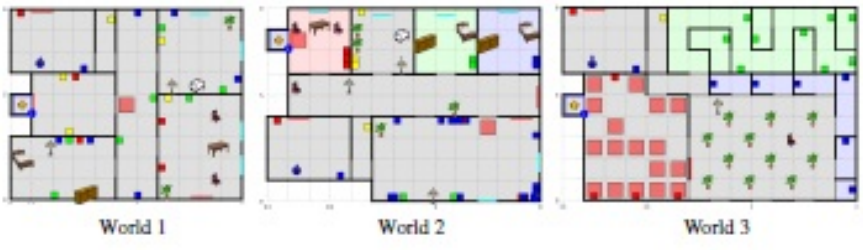


Figure 1: The layout of the 3 worlds in the GIVE-2.5 evaluation world [9].

We are interested in the Potsdam NLG systems [3] because they use AI planning technologies for NLG. However, there are still a lot of things that can be tried. For example, the Potsdam NLG systems use Forward Search (FF). Their outputs provide just one level of abstraction. So, if hierarchical planning is used, we expect the output instructions to be more structured that will lead to the better instruction understanding.

2 Technical Approach

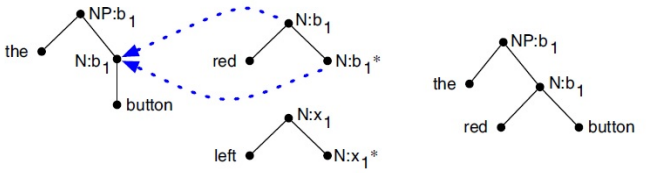
¡¡¡¡¡¡¡ HEAD The Potsdam NLG system [2, 3, 4] is based on the previous work, the CRISP system [6], which is an approach to generate natural language as a planning problem. the CRISP system can generate individual noun phrases. Their follow up work [1] extends the functionality to be able to generate an entire discourse. ======= The Potsdam NLG system [2, 3, 4] is based on the previous work, the CRISP system [6], which is an approach to generate natural language as a planning problem. The CRISP system can generate individual noun phrases. ¿¿¿¿¿¿¿ origin/master

Their contributions are by transforming the tree-adjoining grammar (TAG) into the Planning Domain Definition Language (PDDL). Then, they use fast forward search [5] to generate a dialog plan which will output a sequence of lexicons according to preconditions from both linguistics and non-linguistics context. The lexicons will be organized by an off-the-shelf planner into correct and distinguishing descriptions of the objects in the context. Figure. 2 depicts an example of the CRISP system that converts the TAG grammar to the planning problem for a phrase 'the red button'.

For evaluation, there are manual evaluation and automatic evaluation [4]. The manual evaluation will try to measure how well the users follow the instructions as IF which may be in term of success rate or how fast the users can solve the task. However, our project is small so we lean to automatic evaluation. [4] proposed a maximum entropy model for automatic evaluation which models the success rate $P(succ(r) = 1|r, s)$ from the attribute types in the sentence $a_j(r)$ according to a referential scene $c_i(s)$.

$$P(succ(r) = 1|r, s) = \frac{1}{e^{\sum_j (v_j(s) * a_j(r) + w_0)} + 1}, \tag{1}$$

where $v_j(s) = \sum_i w_{ij} * c_i(s)$. The weights w_{ij} are estimated from the corpus. Reusing the trained model may be a good choice.



(a) TAG grammar for 'the red button' which refers to the button 'b1'.

red(u, x):
Precond: $\mathsf{canadjoin}(\mathsf{N}, u), \mathsf{referent}(u, x), \mathsf{red}(x), \ldots$
Effect: $\forall y.\big(\neg\mathsf{red}(y) \rightarrow \neg\mathsf{distractor}(u, y)\big), \ldots$

left(u, x):
Precond: $\forall y.\neg\big(\mathsf{distractor}(u, y) \wedge \mathsf{left\text{–}of}(y, x)\big)$,
$\mathsf{canadjoin}(\mathsf{N}, u), \mathsf{referent}(u, x), \ldots$
Effect: $\forall y.\big(\mathsf{left\text{–}of}(x, y) \rightarrow \neg\mathsf{distractor}(u, y)\big), \ldots$

the–button(u, x):
Precond: $\mathsf{subst}(\mathsf{NP}, u), \mathsf{referent}(u, x), \mathsf{button}(x), \ldots$
Effect: $\forall y.\big(\neg\mathsf{button}(y) \rightarrow \neg\mathsf{distractor}(u, y)\big), \neg\mathsf{subst}(\mathsf{NP}, u), \ldots$

(b) The planning problem as preconditions and effects of lexicons 'red', 'left' and 'the-button'.

Figure 2: An example of TAG, preconditions and postconditions for some lexicons [4].

3 Project Management

Task to Accomplish:

- Setup the client, NLG server and the matchmaker.
- Understand their NLG system implementation.
- Make a hierarchical planner version of their NLG system.
- Compare the results.

Timeline:

March
Setup the client, NLG server and the matchmaker.

March
Understand their NLG system implementation.

April
Make a hierarchical planner version of their NLG system.

April
Compare the results.

May

Write summary paper.

4 Conclusion

We will try to rebuild the GIVE-2.5 challenge evaluation system. We also pick up the Potsdam NLG system which use AI planning techniques. We try to compare hierarchical planner with their conventional fast forward search (FF).

References

[1] Konstantina Garoufi and Alexander Koller*. Automated planning for situated natural language generation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, Uppsala, 2010.

[2] Konstantina Garoufi and Alexander Koller. Combining symbolic and corpus-based approaches for the generation of successful referring expressions. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 121–131. Association for Computational Linguistics, 2011.

[3] Konstantina Garoufi and Alexander Koller. The potsdam nlg systems at the give-2.5 challenge. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 307–311. Association for Computational Linguistics, 2011.

[4] Konstantina Garoufi and Alexander Koller. Generation of effective referring expressions in situated context. *Language, Cognition and Neuroscience*, 29(8):986–1001, 2014.

[5] Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. 14:253–302, 2001.

[6] Alexander Koller and Matthew Stone*. Sentence generation as planning. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 336–343, Prague, 2007.

[7] Alexander Koller, Kristina Striegnitz, Donna Byron, Justine Cassell, Robert Dale, Johanna Moore, and Jon Oberlander. The first challenge on generating instructions in virtual environments. In *Empirical Methods in Natural Language Generation*, pages 328–352. Springer, 2010.

[8] Alexander Koller, Kristina Striegnitz, Andrew Gargett, Donna Byron, Justine Cassell, Robert Dale, Johanna Moore, and Jon Oberlander. Report on the second nlg challenge on generating instructions in virtual environments (give-2). In *Proceedings of the 6th international natural language generation conference*, pages 243–250. Association for Computational Linguistics, 2010.

[9] Kristina Striegnitz, Alexandre Denis, Andrew Gargett, Konstantina Garoufi, Alexander Koller, and Mariët Theune. Report on the second second challenge on generating instructions in virtual environments (give-2.5). In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 270–279. Association for Computational Linguistics, 2011.