The Software Architecture for the First Challenge on Generating Instructions in Virtual Environments

Alexander Koller
Saarland University
koller@mmci.uni-saarland.de

Donna Byron
Northeastern University
dbyron@ccs.neu.edu

Justine Cassell
Northwestern University
justine@northwestern.edu

Robert Dale
Macquarie University
Robert.Dale@mq.edu.au

Johanna Moore
University of Edinburgh
J.Moore@ed.ac.uk

Jon Oberlander
University of Edinburgh
J.Oberlander@ed.ac.uk

Kristina Striegnitz
Union College
striegnk@union.edu

Abstract

The GIVE Challenge is a new Internet-based evaluation effort for natural language generation systems. In this paper, we motivate and describe the software infrastructure that we developed to support this challenge.

1 Introduction

Natural language generation (NLG) systems are notoriously hard to evaluate. On the one hand, simply comparing system outputs to a gold standard is not appropriate because there can be multiple generated outputs that are equally good, and finding metrics that account for this variability and produce results consistent with human judgments and task performance measures is difficult (Belz and Gatt, 2008; Stent et al., 2005; Foster, 2008). On the other hand, lab-based evaluations with human subjects to assess each aspect of the system's functionality are expensive and time-consuming. These characteristics make it hard to compare different systems and measure progress.

GIVE ("Generating Instructions in Virtual Environments") (Koller et al., 2007) is a research challenge for the NLG community designed to provide a new approach to NLG system evaluation. In the GIVE scenario, users try to solve a treasure hunt in a virtual 3D world that they have not seen before. The computer has a complete symbolic representation of the virtual environment. The challenge for the NLG system is to generate, in real time, natural-language instructions that will guide the users to the successful completion of their task (see Fig. 1). One crucial advantage of this generation task is that the NLG system and the user can be physically separated. This makes it possible to carry out a task-based evaluation over the Internet – an approach that has been shown to provide generous amounts of data in earlier studies (von Ahn and Dabbish, 2004; Orkin and Roy, 2007).

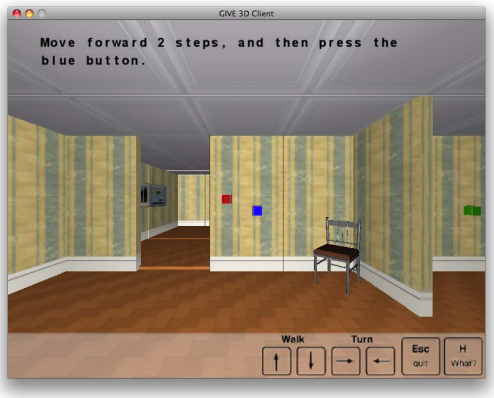

Figure 1: The GIVE Challenge.

In this paper, we describe the software architecture underlying the GIVE Challenge. The software connects each player in a 3D game world with an NLG system over the Internet. It is implemented and open source, and can be a used online during EACL at www.give-challenge.org. In Section 2, we give an introduction to the GIVE evaluation methodology by describing the experience of a user participating in the evaluation, the nature of the data we collect, and our scientific goals. Then we explain the software architecture behind the scenes and sketch the API that concrete NLG systems must implement in Section 3. In Section 4, we present some preliminary evaluation results, before we conclude in Section 5.

2 Evaluation method

Users participating in the GIVE evaluation start the 3D game from our website at www.give-challenge.org. They then see a 3D game window as in Fig. 1, which displays instructions and allows them to move around in the world and manipulate objects. The first room is a tutorial room where users learn how to interact with

Proceedings of the EACL 2009 Demonstrations Session, pages 33–36,
Athens, Greece, 3 April 2009.

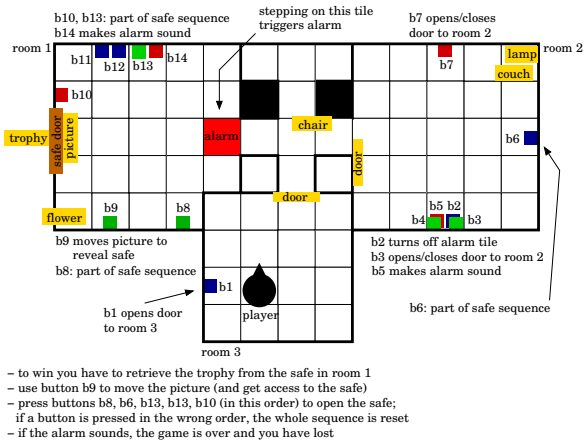

Figure 2: The map of a virtual world.

the system; they then enter one of three evaluation worlds, where instructions for solving the treasure hunt are generated by an NLG system.

The map of one of the game worlds is shown in Fig. 2: In this world, players must pick up a trophy, which is in a wall safe behind a picture. In order to access the trophy, they must first push a button to move the picture to the side, and then push another sequence of buttons to open the safe. One floor tile is alarmed, and players lose the game if they step on this tile without deactivating the alarm first. There are also a number of distractor buttons which either do nothing when pressed or set off an alarm. These distractor buttons are intended to make the game harder and, more importantly, to require appropriate reference to objects in the game world. Finally, game worlds can contain a number of objects such as chairs and flowers which are irrelevant for the task, but can be used as landmarks by a generation system.

Users are asked to fill out a before- and after-game questionnaire that collects some demographic data and asks the user to rate various aspects of the instructions they received. Every action that players take in a game world, and every instruction that a generation system generates for them, is recorded in a database. In addition to the questionnaire data, we are thus able to compute a number of objective measures such as:

- the percentage of users each system leads to a successful completion of the task;
- the average time, the average number of instructions, and the average number of in-game actions that this success requires;
- the percentage of generated referring expressions that the user resolves correctly; and
- average reaction times to instructions.

It is important to note that we have designed the GIVE Challenge not as a competition, but as a friendly evaluation effort where people try to learn from each other's successes. This is reflected in the evaluation measures above, which are in tension with one another: For instance, a system which gives very low-level instructions ("move forward"; "ok, now move forward"; "ok, now turn left") will enjoy short reaction times, but it will require more instructions than a system that aggregates these. To further emphasize this perspective, we will also provide a number of diagnostic tools, such as heat maps that show how much time users spent on each tile, or a playback function which displays an entire game run in real time.

In summary, the GIVE Challenge is a novel evaluation effort for NLG systems. It is motivated by real applications (such as pedestrian navigation and the generation of task instructions), makes no assumptions about the internal structure of an NLG system, and emphasizes the *situated* generation of discourse in a simulated physical environment. The game world is scalable; it can be made more complex and it can be adapted to focus on specific issues in natural language generation.

3 Architecture

A crucial aspect of the GIVE evaluation methodology is that it physically separates the user and the NLG system and connects them over the Internet. To achieve this, the GIVE software infrastructure consists of three components:

1. the *client*, which displays the 3D world to users and allows them to interact with it;
2. the *NLG servers*, which generate the natural-language instructions; and
3. the *Matchmaker*, which establishes connections between clients and NLG servers.

These three components run on different machines. The client is downloaded by users from our website and run on their local machine; each NLG server is run on a server at the institution that implemented it; and the Matchmaker runs on a central server we provide.

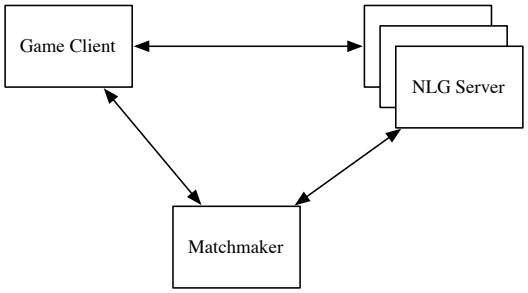

Figure 3: The GIVE architecture.

```
abstract class NlgSystem:
    void connectionEstablished();
    void connectionDisconnected();
    void handleStatusInformation(Position playerPosition,
                                 Orientation playerOrientation,
                                 List⟨String⟩ visibleObjects);
    void handleAction(Atom actionInstance,
                      List⟨Formula⟩ updates);
    void handleDidNotUnderstand();
    void handleMoveTurnAction(Direction direction);
    ...
```

Figure 4: The interface of an NLG system.

When a user starts the client, it connects over the Internet to the Matchmaker. The Matchmaker then selects a game world and an NLG server at random, and requests the NLG server to spawn a new *server instance*. It then sends the game world to the client and the server instance and disconnects from them, ready to handle new connections from other clients. The client and the server instance play one game together: Whenever the user does something, the client sends a message about this to the server instance, and the server instance can also send a message back to the client at any time, which will then be displayed as an instruction. When the game ends, the client and the server instance disconnect from each other. The server instance sends a log of all game events to the Matchmaker, and the client sends the questionnaire results to the Matchmaker; these then are stored in the database for later analysis.

All of these components are implemented in Java. This allows the client to be portable across all major operating systems, and to be started directly from the website via Java Web Start without the need for software installation. We felt it was important to make startup of the client as effortless as possible, in order to maximize the number of users willing to play the game. Unsurprisingly, we had to spend the majority of the programming time on the 3D graphics (based on the free jMonkeyEngine library) and the networking code. We could have reduced the effort required for these programming tasks by building upon an existing virtual 3D world system such as Second Life. However, we judged that the effort needed to adapt such a system to our needs would have been at least as high (in particular, we would have had to ensure that the user could only move according to the rules of the GIVE game and to instrument the virtual world to obtain real-time updates about events), and the result would have been less extensible to future installments of the challenge.

Since we provided all the 3D, networking, and database code, the research teams being evaluated were able to concentrate on the development of their NLG systems. Our only requirement was that they implement a concrete subclass of the class `NlgSystem`, shown in Fig. 4. This involves overriding the six abstract callback methods in this class with concrete implementations in which the NLG system reacts to specific events. The methods `connectionEstablished` and `connectionDisconnected` are called when users enter the game world and when they disconnect from the game. The method `handleAction` gets called whenever the user performs some physical action, such as pushing a button, and specifies what changed in the world due to this action; `handleMoveTurnAction` gets called whenever the user moves; `handleDidNotUnderstand` gets called whenever users press the H key to signal that they didn't understand the previous instruction; and `handleStatusInformation` gets called once per second and after each user action to inform the server of the player's position and orientation and the visible objects. Ultimately, each of these method calls gets triggered by a message that the client sends over the network in reaction to some event; but this is completely hidden from the NLG system developer.

The NLG system can use the method `send` to send a string to the client to be displayed. It also has access to various methods querying the state of the game world and to an interface to an external planner which can compute a sequence of actions leading to the goal.

4 First results

For this first installment of the GIVE Challenge, four research teams from the US, the Netherlands,

and Spain provided generation systems, and a number of other research groups expressed their interest in participating, but weren't able to participate due to time constraints. Given that this was the first time we organized this task, we find this a very encouraging number. All four of the teams consisted primarily of students who implemented the NLG systems over the Northern-hemisphere summer. This is in line with our goal of taking this first iteration as a "dry run" in which we could fine-tune the software, learn about the easy and hard aspects of the challenge, and validate the evaluation methodology.

Public involvement in the GIVE Challenge was launched with a press release in early November 2008; the Matchmaker and the NLG servers were then kept running until late January 2009. During this time, online users played over 1100 games, which translates into roughly 75 game runs for each experimental condition (i.e., five different NLG systems paired with three different game worlds). To our knowledge, this makes GIVE the largest NLG evaluation effort yet in terms of experimental subjects.

While we have not yet carried out the detailed evaluation, the preliminary results look promising: a casual inspection shows that there are considerable differences in task success rate among the different systems.

While there is growing evidence from different research areas that the results of Internet-based evaluations are consistent with more traditional lab-based experiments (e.g., (Keller et al., 2008; Gosling et al., 2004)), the issue is not yet settled. Therefore, we are currently conducting a lab-based evaluation of the GIVE NLG systems, and will compare those results to the qualitative and quantitative data provided by the online subjects.

5 Conclusion

In this paper, we have sketched the GIVE Challenge and the software infrastructure we have developed for it. The GIVE Challenge is, to the best of our knowledge, the largest-scale NLG evaluation effort with human experimental subjects. This is made possible by connecting users and NLG systems over the Internet; we collect evaluation data automatically and unobtrusively while the user simply plays a 3D game. While we will report on the results of the evaluation in more detail at a later time, first results seem encouraging in that the performance of different NLG systems differs considerably.

In the future, we will extend the GIVE Challenge to harder tasks. Possibilities includ making GIVE into a dialogue challenge by allowing the user to speak as well as act in the world; running the challenge in a continuous world rather than a world that only allows discrete movements; or making it multimodal by allowing the NLG system to generate arrows or virtual human gestures. All these changes would only require limited changes to the GIVE software architecture. However, the exact nature of future directions remains to be discussed with the community.

References

A. Belz and A. Gatt. 2008. Intrinsic vs. extrinsic evaluation measures for referring expression generation. In *Proceedings of ACL-08:HLT, Short Papers*, pages 197–200, Columbus, Ohio.

M. E. Foster. 2008. Automated metrics that agree with human judgements on generated output for an embodied conversational agent. In *Proceedings of INLG 2008*, pages 95–103, Salt Fork, OH.

S. D. Gosling, S. Vazire, S. Srivastava, and O. P. John. 2004. Should we trust Web-based studies? A comparative analysis of six preconceptions about Internet questionnaires. *American Psychologist*, 59:93–104.

F. Keller, S. Gunasekharan, N. Mayo, and M. Corley. 2008. Timing accuracy of web experiments: A case study using the WebExp software package. *Behavior Research Methods*, to appear.

A. Koller, J. Moore, B. di Eugenio, J. Lester, L. Stoia, D. Byron, J. Oberlander, and K. Striegnitz. 2007. Shared task proposal: Instruction giving in virtual worlds. In M. White and R. Dale, editors, *Working group reports of the Workshop on Shared Tasks and Comparative Evaluation in Natural Language Generation*. Available at `http://www.ling.ohio-state.edu/nlgeval07/report.html`.

J. Orkin and D. Roy. 2007. The restaurant game: Learning social behavior and language from thousands of players online. *Journal of Game Development*, 3(1):39–60.

A. Stent, M. Marge, and M. Singhai. 2005. Evaluating evaluation methods for generation in the presence of variation. In *Proceedings of CICLing 2005*.

L. von Ahn and L. Dabbish. 2004. Labeling images with a computer game. In *Proceedings of the ACM CHI Conference*.