A Collaborative Puzzle Game to Study Situated Dialog

Andrew Danise and **Kristina Striegnitz**
Union College
Schenectady, NY, USA
danisea@garnet.union.edu, striegnk@union.edu

Abstract

This paper describes a two-player collaborative 2D puzzle game, designed to elicit task-oriented situated dialog. It is going to be used to collect corpora of human-human interactions consisting of language interleaved with actions in the game world and to evaluate natural language generation systems replacing one of the players.

Introduction

The development of a natural language processing (NLP) applications often requires human data. In particular, corpora of human language in the target domain are an essential resource for designing, training, and evaluating NLP systems, and task based evaluations where human users interact with an NLP system give the most realistic assessment of the system. Both corpus production and task based evaluations are expensive and time-consuming, with one major factor being the recruiting of participants.

Recently, a number of projects have used online games to recruit people to help with the collection and annotation of corpora and the task-based evaluation of NLP applications. For example, in the Restaurant Game[1] (Orkin and Roy 2007) players assume the roles of a waitress and a customer in a restaurant. The resulting corpus of human language and actions in this scenario was used to automate the construction of a conversational character that could play one of these roles.

Chamberlain, Poesio, and Kruschwitz (2008) crowdsource the annotation of anaphoric expressions in texts. Players/annotators in the game Phrase Detective[2] collect points for creating or validating annotations. They receive titles when passing certain thresholds and can compare themselves to other players on a leaderboard.

The GIVE Challenge[3] (Koller et al. 2010; Striegnitz et al. 2011) invited players to find a trophy in a virtual environment by following automatically generated instructions. Players were paired with natural language generation systems supplied by different research teams and the collected data was used to evaluate and compare the systems.

This paper describes a puzzle game in which two players collaborate to push one or more balls into a goal position. The players are in two different 2D-environments, but they can drop blocks into the other environment, which are necessary to direct the ball. Furthermore, there are portals which allow the balls to pass back and forth between the environments. The players may or may not be able to see their partner's environment, but they can communicate using a chat interface.

With this game we want to expand on the research started in the GIVE project. As in GIVE, we are interested in the way humans interleave language and actions when they are situated in an environment. This is a topic that has recently started to receive increased attention in the natural language generation and dialog systems communities, e.g. (Stoia et al. 2006; Garoufi and Koller 2010; Dethlefs, Cuayáhuitl, and Viethen 2011). By manipulating whether or not the players can see each other's environment, we want to compare the effects of a shared environment with situations in which the players receive only very indirect information about their partner's environment (through the locations at which blocks are dropped).

In GIVE there are two distinct roles – the instruction giver, who can send messages to the instruction follower but not act in the environment, and the instruction follower, who can act in the environment but not respond by sending messages. In the game described in this paper, we want to create a more balanced scenario in which both players can act as well as contribute to the chat conversation. Furthermore, we want both players to sometime be in the role of the instruction giver and sometimes in the role of the instruction follower.

Finally, we want the game to be fun so that people play multiple levels, return to play again, and tell their friends about it. The feedback we received from participants in the GIVE evaluations indicates that while many players appreciated GIVE as a research project, many others players were disappointed. They came to it expecting a game but then discovered that all they had to do was follow instructions. There were no puzzles to solve for the instruction follower and no creative way to contribute to the solution of the task.

[1]http://web.media.mit.edu/ jorkin/restaurant

[2]http://anawiki.essex.ac.uk/phrasedetectives

[3]Generating Instructions in Virtual Environments; http://give-challenge.org/research

In fact, a significant number of players quit the game before finishing, and most of the games that are canceled or lost end quickly.

Game Design

Our goal in designing this game is to elicit problem solving dialogues between two players who are situated in a virtual environment. We want both players to equally contribute to the dialog as well as the problem solving process. Ultimately, we want to build dialog systems that can take the role of one of the players. That means that the environment has to be simple enough that we can use automated planning techniques to calculate strategies for solving the puzzles.

In this two-player game, each player is situated in a maze created from obstacles and traps of various kinds. Figure 1 illustrates what the game interface may look like for each player at some point in the game[4]. The red block is player one's avatar, the green block player two's avatar. The purple block is the ball, which needs to be pushed to the goal represented by the orange block. The gray blocks are portals. Portals come in pairs – if the ball or player enter one portal, they exit through another one — and connect different areas of one player's environment or link one player's environment with the other's. The light gray portals are open to both the ball and the player; the dark gray portals can only be used by the ball.

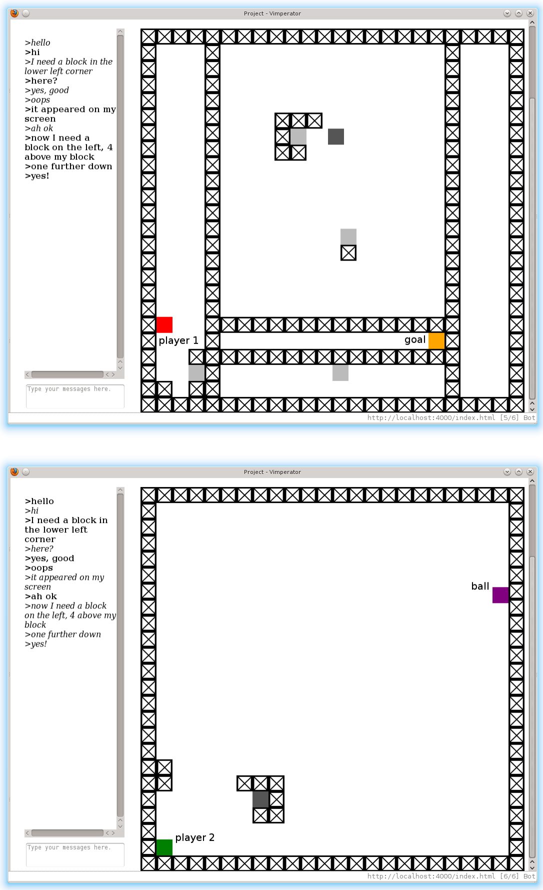
The players can move their avatars using the arrow keys on their keyboard. When they push the ball, it starts moving in the direction of the push and only stops when it collides with an obstacle (see Figure 2(a)). If the ball cannot move in the direction of the push (e.g. because there is an obstacle), it (randomly) picks a direction that is free of obstacles to move to, as illustrated in Figures 2(b) and (c). This behavior makes sure that the ball only gets trapped if it is surrounded by walls on three sides (Figure 2(d)).

Pressing the Ctrl-key allows players to place an obstacle into their partner's environment at the position that corresponds to the player's current location. Each player can only place three blocks on their partner's screen at a time. When the fourth block is placed, the block that was placed first is deleted, as shown in Figure 3.

The players cannot place obstacles into their own environments, and since obstacles are the only way to stop a ball that is moving, they have to collaborate to control the balls movements. The environments are designed to force the players to work together and communicate with each other. The players can use a chat area next to their game environment to coordinate their strategy and to give instructions on where obstacles need to be placed.

We are creating three modes of the game by manipulating whether the players can see their partner's environment, in order to study the effect that additional shared information about the context has on communication. In the first mode, the players don't see their partner's environment (as in Figure 1), but the location of obstacles that get dropped into

[4]While the game is fully playable, the interface is still a prototype in which everything is represented by differently colored blocks.


>hello
>hi
>I need a block in the lower left corner
>here?
>yes, good
>oops
>it appeared on my screen
>ah ok
>now I need a block on the left, 4 above my block
>one further down
>yes!

>hello
>hi
>I need a block in the lower left corner
>here?
>yes, good
>oops
>it appeared on my screen
>ah ok
>now I need a block on the left, 4 above my block
>one further down
>yes!

Figure 1: Sample game screens for players one and two.

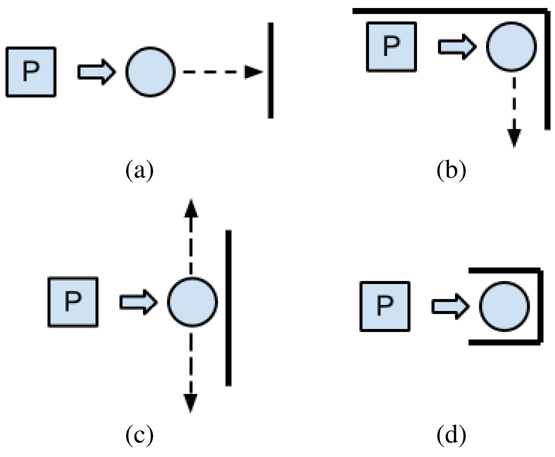


Figure 2: Behavior of the ball.

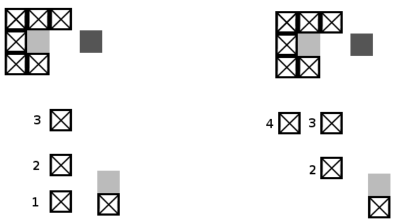


Figure 3: Each player can place at most three obstacles at a time. When the fourth obstacle gets placed, the first one disappears.

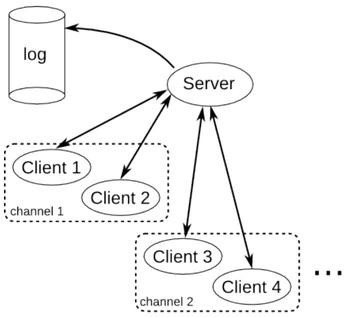


Figure 4: Structure of the server-client relationship.

their environment give them some hints where their partner is. In mode two, the players can see a shadow of their partner's avatar moving around their own environment, so that they always know where their partner is, but they don't have any information about the layout of their partner's environment. And in the third mode, the players see their own environment and their partner's side by side.

The game starts out with relatively simple environments, but gets more complicated as additional kinds of obstacles and traps and more complex variations of the task get introduced. For example, there may be pits which can swallow the ball (or player), bouncy obstacles which reflect the ball when it hits them, moving fireballs which burn and kill the ball or player, destructible barriers which the partner can explode by dropping an obstacle onto them, or forking portals which send the ball/player randomly to one of two exits. At some point in the game, each player is given their own set of one or more balls. This will allow for some interesting variations on the task. For example, there may be multiple goal positions and the players have to navigate their balls to corresponding goals. Or the balls are numbered (e.g. player one has balls number 1, 3 and 4 and player two has balls 2 and 5) and the balls have to be push into goals in order. Or the balls have colors and both players have to move a ball of the same color to a goal before they can work on the next ball with a different color.

Implementation

The game is implemented using a client-server architecture illustrated in Figure 4. The client-server interaction is as follows:

1. A player requests the game's webpage. The http server sends all game files.
2. The game's client-server connection is established.
3. The server assigns the client either to a channel containing only one player or a new channel.
4. Once two clients are in a channel, the server sends the signal to start to the clients.
5. The server listens and passes on any messages sent between players.
6. The server logs all movements in the game world and the communication between the two players.

Since we would like to use this game to collect human-human conversations and evaluate dialog systems over the Internet, it should be as easy as possible to run the client. Therefore, we decided to develop a browser based-game. Both the client and the server are written in JavaScript.

The client uses Crafty[5], a JavaScript and HTML5 game engine, which facilitates drawing the game area into the browser window, event management and collision detection. In the beginning of the project, we explored a number of JavaScript game engines and settled on Crafty because it seemed to satisfy our needs, was under active development and came with many tutorials and good documentation.

The server is built on Node.js[6], a server-side JavaScript framework. In particular, the express module[7] is used to create an HTTP server, and the in-game client-server communication is implemented using Socket.IO[8].

Conclusions and Outlook

This paper has described a collaborative two-player puzzle game which we have designed as a tool for collecting human-human problem-solving dialogs in a situated scenario. We have implemented a prototype and are currently designing and playtesting more levels. Once we have created a set of levels of varying difficulty, we will start collecting human-human interactions, first in the lab at Union College and then by making it available online. We plan to use the collected data to develop a conversational system that can play this game and to evaluate it over the Internet.

The motivation for this research is to learn more about communication in situated environments where language and actions get interleaved and to develop algorithms for effective communication strategies that can be implemented in dialog systems and transfer to other virtual environments and tasks.

References

Chamberlain, J.; Poesio, M.; and Kruschwitz, U. 2008. Phrase Detectives: A web-based collaborative annotation game. In *Proceedings of the International Conference on Semantic Systems (I-Semantics08).*

[5]http://craftyjs.com
[6]http://nodejs.org/
[7]http://expressjs.com/
[8]http://socket.io

Dethlefs, N.; Cuayáhuitl, H.; and Viethen, J. 2011. Optimising natural language generation decision making for situated dialogue. In *Proceedings of the SIGDIAL 2011 Conference*, 78–87.

Garoufi, K., and Koller, A. 2010. Automated planning for situated natural language generation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 1573–1582.

Koller, A.; Striegnitz, K.; Byron, D.; Cassell, J.; Dale, R.; Moore, J.; and Oberlander, J. 2010. The First Challenge on Generating Instructions in Virtual Environments. In Krahmer, E., and Theune, M., eds., *Empirical Methods in Natural Language Generation*, volume 5790 of *LNCS*. Springer. 337–361.

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

Stoia, L.; Byron, D. K.; Shockley, D.; and Fosler-Lussier, E. 2006. Sentence planning for realtime navigational instruction. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, 157–160.

Striegnitz, K.; Denis, A.; Gargett, A.; Garoufi, K.; Koller, A.; and Theune, M. 2011. Report on the *Second* Second Challenge on Generating Instructions in Virtual Environments (GIVE-2.5). In *Proceedings of the Generation Challenges Session at the 13th European Workshop on Natural Language Generation*, 270–279.