

Experiments in Paired Storytelling: 2023 Dataset Description

Introduction

This document describes a dataset was collected from an interactive narrative game played by 18 University of Kentucky students in an undergraduate game development class exercise. Not all game sessions from the exercise are included in the dataset; consent forms were offered to the participants, and the logs for any session with a non-consent-form-signing participant were deleted. The dataset includes 509 actions across 22 game sessions. The next two sections describe the game itself to provide context for the data, and the last section describes the format of the dataset itself in this archive.

Game Interface

Each game session had two users, one taking the role of the *game master* and one taking the role of the *player*. Both users played through a browser interface like the one in the image below.

State

Player
Location: Market
Status: alive
Inventory: Potion

Bandit
Location: Crossroads
Status: alive
Inventory: Dagger

Guard
Location: Market
Status: alive
Inventory: Halberd
Can receive crime reports.

Merchant
Location: Market
Status: alive
Inventory: Sword, Coin
Can initiate or receive trade proposals.

Grandma
Location: Cottage
Status: comatose
Inventory:
Has no actions available.

Chest
Location: Camp
Inventory: Jewel

Game

You are the Game Master (GM). Your partner controls the Player character; you control the rest.
The Bandit NPC is at their camp plotting their next acquisitions.
The Guard NPC is on duty at the market.
The Merchant NPC is peddling wares at the market.
The player character's Grandma is too sick to do anything but lie in bed.
Player goes to the Crossroads.
Player goes to the Market.
Merchant offers Potion to Player in exchange for a Coin.
Merchant exchanges Potion for Player's Coin.
Bandit goes to the Crossroads.

Choose a character to act. (The player is paused and can act only if you pass control.)

- [Bandit](#)
- [Guard](#)
- [Merchant](#)
- [Let the player act](#)

Rate

Agency:
I can have meaningfully different experiences depending on my own choices.
★★★★★

Structure:
These events feel like a story, rather than a random sequence.
★★★★★

The left pane showed the properties of game objects such as characters; for the game master, this included all game information, while for the player, it was restricted to objects in the player character's location.

The middle pane showed game narration and a menu for selecting *actions*. Actions were the means by which properties of game objects were updated and new narration was produced. Only one user had action menus available at any given time; by default, the menus were available to the game master, who could allow the player to choose actions by selecting "Let the player act".

The right pane had a tool for player to submit ratings, between 1 and 5 stars, for two qualities of the story, *structure* and *agency*.

After the game session ended, each user was given a post-game interview where they were asked to explain their own and their partner's action choices one-by-one as illustrated below.

Review the story and answer these questions.

- Player goes to the Crossroads.
- Player goes to the Market.
- Merchant offers Potion to Player in exchange for a Coin.

Why do you think your partner chose this action?

Game Mechanics

This section explains the actions that appear in the game logs. Each action operated on game objects from among the following types.

- **Locations** have adjacent locations.
 - The **Crossroads** location is adjacent to all other locations.
 - The **Camp**, **Cottage**, and **Market** locations are adjacent only to the Crossroads.
- **Characters** have a controlling user from among the player and game master; a current status from among *alive* and *dead*; and a current location.
 - The **Player** character is controlled by the player and starts alive at the Cottage.
 - The **Bandit** character is controlled by the game master and starts alive at the Camp.
 - The **Guard** character is controlled by the game master and starts alive at the Camp.
 - The **Merchant** character is controlled by the game master and starts alive at the Camp.
 - The **Grandma** character is at the Cottage with a special *comatose* status and does not participate in actions or otherwise have a functional effect on the game state.
- **Containers** have a location.
 - The **Chest** container is at the Camp.
- **Items** have a possessing character or container. **Weapons** are a subtype.
 - The **Coin** item starts with the Player.
 - The **Dagger** weapon starts with the Bandit.
 - The **Halberd** weapon starts with the Guard.
 - The **Jewel** item starts in the Chest.
 - The **Potion** item and **Sword** weapon start with the Merchant.

Although actions were communicated to users in the form of natural-language flavor text, they are represented in the logs as *action signatures* consisting of the action name and list of parameters. The actions in the game were as follows:

- **attack([character], [character], [weapon or None]):** Available to the controller of the first character when both characters are alive at the same location and the first character has the weapon (if applicable). The choice or lack of weapon has no functional effect on the game state and affects only the narration text. The game master chooses the outcome:
 - **attackKill:** The second character becomes dead.
 - **attackMiss:** The action has no effect.
- **badEnding():** Available to the game master when the Player character is dead. The game ends.
- **goodEnding():** Available to the game master when the Player character is at the Cottage and has the Potion. The game ends.
- **loot([character], [character or container], [item]):** Available to the controller of the first character when the first character is alive, the other character or container is at the same location and has the item, and the other character (if applicable) is dead. The first character gains the item.
- **report([character], [character], [location]):** Available to the controller of the first character when the first character and the Guard character are alive at the same location. Has no functional effect on the game state; only displays flavor text that the first character tells the Guard that the second character has committed a crime at the specified location.
- **rob([character], [character], [item], [weapon or None]):** Available to the controller of the first character when both characters are alive at the same location, the second character has the item, the first character has the weapon (if applicable). The choice or lack of weapon has no functional effect on the game state and affects only the narration text. The controller of the second character chooses from the following:
 - **robSuccess:** The first character gains the item.
 - The choice passes to the game master who selects from the following outcomes:
 - **robDie:** The second character dies.
 - **robEscape:** The action has no effect.
 - **robForced:** The first character gains the item.
- **tradeOffer([character], [character], [item], [item]):** Available to the controller of the first character when both characters are alive at the same location, one of the

characters is the Merchant and the first character has the first item. The controller of the second character chooses from the following:

- **tradeAccept:** The first character gains the second item and the second character gains the first item. Available only when the second character has the second item.
- **tradeDecline:** The action has no effect.
- **tradeInterest:** The action has no effect but the flavor text indicates the second character's desire for the trade. Available only when the second character does not have the second item.
- **travel([character], [location]):** Available to the controller of the character when the character is at an adjacent location. The character's location becomes the chosen location.

Data Contents

Each time a user took an action in the game session, a corresponding row was added to the game logs in the dataset. Entries in *original.csv*, after an initial header row, have the following columns in order:

- **sessionID:** Assigned uniquely to each game session.
- **eventID:** Within a game session, assigned uniquely to an instance of an in-game action; i.e., each combination of a *sessionID* and an *eventID* is unique across the whole dataset.
- **timestamp:** A timestamp showing the *minutes:seconds* after the start of the class session at 11am Eastern on November 21, 2023.
- **gmID:** The user ID of the session's game master.
- **playerID:** The user ID of the session's player.
- **action:** The action signature for the action taken.
- **userWhoChose:** The user ID of the user who selected the action (either the *gmID* or the *playerID*).
- **outcome:** The result if the current action prompted further choices (e.g., the decision to accept or decline a trade offer). Blank if no such choices were prompted.
- **gmExplanation:** The game master's response to the post-game interview question about the current action. Blank if the user was not prompted to explain the action.
- **playerExplanation:** The player's response to the post-game interview question about the current action. Blank if the user was not prompted to explain the action.

- **gmStructureRating:** The game master’s star rating for structure, if the game master submitted a rating after the current action and before any other action. Blank if the user did not submit a rating during that timeframe.
- **gmAgencyRating:** The game master’s star rating for agency, if the game master submitted a rating after the current action and before any other action. Blank if the user did not submit a rating during that timeframe.
- **playerStructureRating:** The player’s star rating for structure, if the player submitted a rating after the current action and before any other action. Blank if the user did not submit a rating during that timeframe.
- **playerAgencyRating:** The player’s star rating for agency, if the player submitted a rating after the current action and before any other action. Blank if the user did not submit a rating during that timeframe.
- **finalGMStructure:** The game master’s rating for structure at the end of the entire game session. Within a single game session, all entries for this column are the same.
- **finalGMAgency:** The game master’s rating for agency at the end of the entire game session. Within a single game session, all entries for this column are the same.
- **finalPlayerStructure:** The player’s rating for structure at the end of the entire game session. Within a single game session, all entries for this column are the same.
- **finalPlayerAgency:** The player’s rating for agency at the end of the entire game session. Within a single game session, all entries for this column are the same.

The *labeled.csv* file contains additional data that we generated for our analysis rather than being directly part of the data collected from the users. Entries have the same contents as in *original.csv*, followed by these additional columns in order:

- **labelIntentMatching:** A label assigned manually by the authors. Assesses the agreement between the *gmExplanation* and *playerExplanation*, with higher numbers indicating stronger agreement. Blank if the explanations were not elicited for the current action. See Section 5.2 of the INT publication for a full explanation.
- **causallyConnectedToEnding:** A label computed as follows: In a Sabre-based representation of the game session (see below), we determined whether the current action is part of a sequence of events that enable each other such that the ending of the story is the final event on the chain. The label is *TRUE* if so, *FALSE* otherwise, or blank for the final event itself. See Section 5.5 of the INT publication for a full explanation.
- **gmInterventionTaken:** A label computed as follows: If the current action is a player action and the game master chose the *outcome* such that the action had no effect,

the label is *TRUE*. If the outcome is a player action and the game master *could have* made the action have no effect, but instead the game master allowed the action to have an effect, the label is *FALSE*. Blank otherwise. See Section 5.3 of the INT publication for a full explanation.

The *sabre* folder contains files used in the analysis to represent game logs as plans in the Sabre narrative planner: *gramma-game.txt* is a Sabre problem file that represents the game flow as planning operators. Each file in the *solutions* subfolder corresponds to one game session in the dataset and represents the user decisions from the game session as a Sabre-readable plan.