A Framework for Summarizing Game Experiences as Narratives

Yun-Gyung Cheong and R. Michael Young

Liquid Narrative Group
Department of Computer Science
North Carolina State University
Raleigh, NC 27695
ycheong@ncsu.edu, young@csc.ncsu.edu

Abstract

Online role playing games may be enjoyed periodically for months or years at a time. Players often log onto the game and play for a short time, and then log off for a long period of time before returning to the game, even though events continue in the persistent game world. Providing a summary of events that have occurred while a player was away from the game would help the player maintain game context. In this paper, we present a computational framework that generates a summary from game log messages.

1. Introduction

In recent years, it has become commonplace for game players to spend increasing amounts of time playing games. The users of single player games such as The Sims, Half-Life 2, and Prince of Persia often play a single game over the course of weeks or months by means of saving the game state when a gaming session is finished and loading the game state upon returning to the game. Furthermore, the players of Massively Multi-player Online Role Playing Games (MMORPGs) such as Lineage, World of Warcraft, and Everquest experience a persistent online virtual world. Players become extremely involved in these virtual worlds for extended periods of time (Griffiths et al., 2003). Thus it would be beneficial to provide players with summaries of important events that occur during their absence.

Research on the automatic summarization of game logs is closely related to research in story summarization (Capus and Touringy, 2003; Lehnert, 1981) and automated commentary generation (Tanaka-Ishii et al., 1998; Voelz et al., 1999; André et al., 2000). However, it is difficult to apply this research directly to game log summarization due to a number of limitations. For instance, story summarization models (Capus and Touringy, 2003; Lehnert, 1981) generally require manually pre-processed information which is not present in game logs. In contrast, commentary systems may be able to process raw game logs, however, their content selection is designed to produce text "in the moment" rather than to produce coherent narrative discourse.

To support the functionality of narrative summarization of game experiences, this paper presents a framework for generating a summary text from game log messages. Our fundamental assumption is that a game log can be used to generate a collection of plans that account for the logs' events. From these plans, a series of salient events can be identified by their causal and temporal relationship to the goal of the plans, as evidenced by psychological research on story recall (Trabasso et al., 1984) and question-answering in the context of stories (Graesser et al., 1991). Our approach involves two distinct tasks: translating a game log into a plan structure and constructing a summary centered on important events in a coherent manner.

2. Framework

This section presents a framework for summarizing game experiences as narratives. Our system consists of two main components: a log analyzer and a skeleton builder. The *log analyzer* takes a game log as input and generates a sequence of actions structured as a plan achieving the given mission of the game or of the game's current level. This plan is then sent as input to the *skeleton builder*, which identifies essential elements of the plan to build a summary of the game's events.

2.1 The Log Analyzer

The log analyzer constructs plan structures from a game log. While the current log messages by commercial games contain information mainly for system administrative purposes such as monitoring network traffic, the needs for readable log formats are under consideration (Garner, 2004). For this research, we assume that the log file input to the log analyzer contains various game events (e.g., user identification, user activities, and team activities). We use the plan structure generated by Crossbow, a hierarchical, partial-order causal link planner of the same type as the Longbow planner (Young et al., 1994). [1] In this article, a plan is represented as a totally ordered series of plan steps, $<s_1, s_2, \ldots, s_n>$ where s_i is an instantiation of a plan operator contained in a plan library L and s_i precedes s_j where $i<j$. A

[1] For brevity, we discuss only a non-hierarchical version of Crossbow here.

plan operator *op* is represented as a tuple <*preconds, effects*> where *preconds* are a set of conditions <c_1, c_2, …, c_n> that must be satisfied for *op* to execute correctly while *effects,* <c_1, c_2, …, c_n>, represent those changes to the world state made as a result of the successful execution of a step instantiated from the plan operator *op*. A causal link is represented as ($s_i \rightarrow s_j$; e_k), notating a plan step s_i's effect e_k is used to satisfy the precondition e_k of s_j. Binding constraint is described as <s_i; ((p_1, c_1) (p_2, c_2))> where a plan step s_i binds constant c_1 for p_1 and c_2 for p_2. We use a totally ordered plan structure under the assumption that the log file is written sequentially.

For us to construct a plan from a game log, the log analyzer requires the initial state and the goal of the game world, and a plan library, and performs the following steps. First, for each log message the system chooses a plan operator in the plan library which has the same name as the action name in the message, and instantiates it with the information contained in the message. The log analyzer uses these instantiated actions as the plan steps of a partial plan, and then fills in its missing plan structure such as causal links to construct a complete plan.

To illustrate the plan building process, we present example log messages following the format specification in (Garner, 2004) as in Figure 1, which can be obtained from a shooting game where each message prefixed by a time stamp. Suppose a game in which the user, who plays the character Azure in the game, was given a mission to construct her military base where her enemy was guarding. Once the game began, she picked up a shotgun, and then the guard Ricker attacked her with his rifle. The attack damaged her, and she dropped her gun. Then she picked up her gun and shot the guard Ricker to death. She constructed her base and completed her mission. With given these messages, the log analyzer first converts the log messages into a sequence of instantiated actions. Next, the system builds a partial plan *P* composed of the first step *INIT*, describing the initial game world as its effects, and the last step *FINISH*, describing the goal state as its preconditions. It then modifies *P* by inserting those instantiated actions between the *INIT* and *FINISH* steps. As the last step, the system establishes causal links for every step in *P*. A causal link is made when a precondition of a plan step is unified with an effect of an earlier plan step. When several events are available for the source step, the log analyzer selects the closest event in time to the destination step in order to restrict the interval that the established causal link can be undone by other actions. For example in Figure 1 where the two 'acquiring' actions at T1 and T6 are available as the source step of the 'killing' action at T8, the latter is chosen. The causal link establishment process starts at *FINISH*, which is the last step of *P*, and plans toward the first step *INIT*. This process continues until all the preconditions of every plan step in *P* are checked. Finally, the complete plan *P* (as the diagram in Figure 2) is sent to the skeleton builder. In Figure 2, only those actions which are causally related to the user's goal are shown.

T1: Player "Azure" acquired "shotgun".
T2: Player "Ricker" attacked player "Riker" with "rifle".
T4: Player "Azure" dropped "shotgun".
T6: Player "Azure" acquired "shotgun".
T7: Player "Azure" attacked player "Riker" with "shotgun".
T8: Player "Azure" killed player "Riker" with "shotgun".
T9: Player "Azure" triggered "construct-base".

Figure 1. A log example

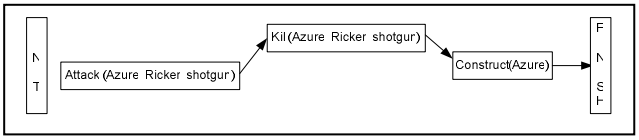

Figure 2. A plan representing a game experience

2.2 The Skeleton Builder

The skeleton builder determines a series of the important actions, the skeleton of the story, based on the user's knowledge. The skeleton builder first generates a candidate skeleton composed of essential events of the plan, and then it tests the skeleton to ensure that its content is coherent, that is, that it can be understood as an integral story.

Our system rates the importance of each event based on a method for extracting important actions that are likely to be included in the story recall, devised by Trabasso et al. (1984). To determine an individual story event's importance, their approach counts the number of causal relationships with other steps in the narrative and measures each event's importance by analyzing its role in a series of actions in a story that are causally related. Adapting their approach, the skeleton builder approximates causal relationships by counting the number of incoming and outgoing causal links of a plan step and measuring the qualitative importance of events which are determined by their roles in the plan. We define three important roles of events in a story plan: an opening act, a closing act, and a motivated act. An opening act is the first action in the plan. A closing act is the last action that occurs in the story. Motivated acts are actions that establish a precondition of the goal state. After computing each event's importance, the top *N* events are identified as kernels. The value for *N* can be set as either an integer or a ratio against the total number of actions in the plan.

Once a skeleton is extracted, the system checks if the skeleton is coherent based on a model of the reader's comprehension process. This model is composed of a reasoning algorithm, a reasoning resource bound, knowledge and preferences. To model a user's reasoning, we use Crossbow, a version of the Longbow planning system (Young et al., 1994). Prior work has provided strong evidence that human task reasoning is closely related to partial-order planning algorithms (Rattermann, 2001) and that refinement search (Kambhampati et al, 1995), the plan construction process performed by Crossbow, can be used as an effective model of the plan reasoning process (Young 1999). The coherency checking

algorithm iterates through two phases: coherency check and event selection. The coherency check phase runs Crossbow to find a complete plan to achieve the goal which contains the events in the skeleton. If such a plan is found, the skeleton is coherent and the algorithm returns. Otherwise, it begins the event selection phase, in which an event not in the skeleton with the highest importance value is added to the skeleton. The algorithm iterates until a complete plan is found. Finally, the system constructs the output text summary as the sequence of log messages that correspond to the actions contained in the skeleton.

2.3. A Summary from an Example Story Plan

This section presents a summary generated by the skeleton builder from a story plan. Figure 3 shows the plan realizes into text where one sentence represents a single action in the plan. In the story, an antagonist Dr. Evil plans to assassinate the President while a poor father Tom plans to trade his ring (which has magic power that Tom isn't aware of) for a toy as a present for his six-year old son Ben. Our system takes this story plan as input and selects #2, #6, #11, #12, and #14 as its summary. The coherency of this summary is not tested in our current implementation.

> [1] Tom traveled to Dr. Evil's castle. [2] Tom traded his ring for Dr. Evil's toy. As a result, Tom obtained the toy that Ben wants to have and Dr. Evil obtained the ring of absolute power. [3] Tom traveled back to his house, and went up to the Christmas tree. [4] Tom put the toy under the Christmas tree. [5] Ben walked from his room to the Christmas tree. [6] Ben found his Christmas present—the toy that Tom left. [7] Dr. Evil went to the Wachovia bank to withdraw money from his bank account. [8] Dr. Evil withdrew enough cash from his account to buy a gun. [9] Dr. Evil traveled to a gun store. [10] Dr. Evil bought a gun. [11] The President invited Dr. Evil to the fund-raising event at the White House. [12] Dr. Evil traveled to the White House. [13] Dr. Evil used the ring of absolute power to put all the Secret Service agents to sleep; as a result, there was no one around the president. [14] Dr. Evil shot the president with his gun and became the ruler of the world.

Figure 3. A story created by Crossbow realized into a text

We carried out a pilot-study to evaluate the quality of this summary with 25 subjects from the North Carolina State University community. The subjects read the input story and then selected a sequence of 5 events that they think the most appropriate as its summary, and then they were asked to compare their summaries with the system-generated summary. The result shows that 32% of the subjects reported that the system generated-summary better represented the story than their own selected ones, and 52% of them answered that the system chosen sentences were as good as theirs; only 12% answered their summaries are better than the system's.

3. Discussion

This paper describes a framework for summarizing gaming experiences as stories by translating a game log into a plan structure and extracting essential events from the plan based on their causal relationship to the goal of the story. A pilot study result supports the claim that our system identifies essential elements of a story consistent with those that a human subject would select. Our future work will augment our plan representation with a semantic level adding domain-specific knowledge to the current skeleton builder to enhance the quality of the summary.

References

André, E., K. Binsted, K. Tanaka-Ishii, S. Luke G. Herzog, and T. Rist. 2000. Three RoboCup Simulation League Commentator Frameworks. *AI Magazine*, 21(1):57-65.

Capus, L. and Tourigny, N. 2003. A case-based reasoning approach to support story summarization. *International Journal of Intelligent Frameworks*, 18, 877-891.

Graesser, A.C., Lang, K.L., & Roberts, R.M. 1991. Question answering in the context of stories. *Journal of Experimental Psychology: General*, 120(3), 254-277.

Garner, S. 2004 June 24. Half-Life Standard Log Format Specification version 1.03. < http://www.hlstats.org/logs/>. Accessed 2006 March 24.

Griffiths, M. D., Davies, M.N.O. & Chappell, D. 2004. Online computer gaming: a comparison of adolescent and adult gamers. *Journal of Adolescence*, 27, 87-96.

Kambhampati, S., Knoblock, C. A., and Yang, Q. 1995. Planning as Refinement Search: A Unified Framework for Evaluating Design Tradeoffs in Partial-Order Planning. *Artificial Intelligence*, 76(1-2), 167-238.

Lehnert, W.G. 1981. Plot units and narrative summarization. *Cognitive Science*, 5(4), 293.

Rattermann, M. J., Spector, L., Grafman, J., Levin, H. and Harward, H. 2002. Partial and total-order planning: evidence from normal and prefrontally damaged populations, *Cognitive Science*, 25(6); 941-975.

Tanaka-Ishii, K., Noda, I., Frank, I., Nakashima, H., Hasida, K., and Matsubara, H. 1998. 'MIKE: An automatic commentary framework for soccer. In *Proceedings of ICMAS*, 285– 292.

Trabasso, T. and Sperry, L. L. 1985. Causal Relatedness and Importance of Story Events. *Journal of Memory and Language*, 24, 595-611.

Voelz, D., André, E., Herzog, G., and Rist, T. 1999. Rocco: A RoboCup Soccer Commentator Framework. M. Asada and H. Kitano (Eds.): RoboCup-98, LNAI 1604, 50-60, Springer-Verlag Heidelberg Berlin.

Young, R.M., Pollack, M.E., and Moore, J.D. 1994. Decomposition and causality in partial-order planning. In *Proceedings of AIPS 1994*, 188-194.

Young, R. M. 1999. Using Grice's Maxim of Quantity to Select the Content of Plan Descriptions, *Artificial Intelligence*, 115(2), 215-256.