Game Usage Mining: Information Gathering for Knowledge Discovery in Massive Multiplayer Games

Amund Tveit*
Department of Computer and Information Science
Norwegian University of Science and Technology

Gisle B. Tveit
Department of Thermal Energy
Norwegian University of Science and Technology

Abstract

Multiplayer games provide rich sources of information for data mining. The primary purpose of data mining in games is to find patterns of behavior, structure or content in order to improve the overall gameplay, hence keeping players longer and increasing the revenue of the game service. In this paper we define the term game usage mining and describe it from web usage mining perspective, with the main emphasis on data gathering. A classification of game types from a data mining perspective is also suggested. Based on the discussion we propose content for a common game log format suitable for game usage mining.

Keywords: Massive Multiplayer Games, Web Usage Mining, Information Gathering

1 Introduction

Multiplayer games, both on the wireless and the traditional Internet, are excellent sources of information for data mining. We can mine usage patterns of both human and virtual players (avatars), or discover patterns in the games content and structure.

*amund.tveit@idi.ntnu.no, IDI/NTNU, N-7491 Trondheim, Norway

In this paper we compare the process of information gathering for behavior mining in massive multiplayer games (game usage mining) to the similar process known from web usage mining.

1.1 Motivation

The main motivation for performing data mining in computer games is to discover patterns, e.g., rules or statistics, that can possibly be used to improve the game. Then paying players become more satisfied and stay longer, which again increases the revenue of the game service. This is of particular importance for wireless games, since keeping a player is equivalent of making more money. Wireless Internet revenue models are either proportional to money per time unit spent by the player (e.g., WAP over GSM), or increasingly more common, money per byte served to the player (e.g., UMTS, I-Mode and WAP over GPRS).

Areas and connections between areas (e.g., doors or tunnels), are frequent building structures in games. From a web usage mining perspective they can be seen as relatively analog to web pages and links (i.e., URLs), respectively. Player behavior can also be detected, including actions and speech act utterances. This makes it possible generate game logs that resemble web logs in structure, and hence can gain from applying web usage mining methods with only minor adoptions.
1.2 Terminology

In order to have a clear vocabulary to describe data mining in multiplayer games (from now on called game mining) we define three main types: 1) game content mining – discovery of patterns in multimedia or textual content in games (e.g. room layout), 2) game structure mining – discovery of structural patterns in form of paths and connections binding the game world together (e.g. hallways between rooms), and 3) game usage mining – discovery of human and avatar behavior patterns. The described types of game mining are inspired by well-known types of web mining – web content mining, web structure mining and web usage mining [2].

1.3 Research Problem

What are the similarities and differences of information gathering for web usage mining and game usage mining?

The rest of this paper is organized as follows. Section 2 describes a game classification schema. Section 3 describes the Game Usage Mining concept. Section 4 compares information gathering in a web and game context. Section 5 describes the proposed game log approach, and finally the conclusion with future work.

2 Game Classification

From a information gathering viewpoint, games are proposed classified according to figure 1, with examples for each class.

<table>
<thead>
<tr>
<th></th>
<th>Singleplayer (SPG)</th>
<th>Multiplayer (MPG)</th>
<th>Massive Multiplayer (MMPG)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete Gamestate</td>
<td>Solitaire, Tetris</td>
<td>Go, Backgammon, Chinese Checkers</td>
<td>?</td>
</tr>
<tr>
<td>Non-Discrete Gamestate</td>
<td>Zelda, The Longest Journey</td>
<td>Quake, MUD/00</td>
<td>Everquest, Lineage</td>
</tr>
</tbody>
</table>

Figure 1: Computer Game Classification

In the discrete game-state class we consider games that have a discrete search space and a turn-based gameplay, e.g. in Chinese Checkers it is not possible to do fractional (non-integer) moves of marbles, and players have to wait until their turn. In Quake [1] the search space (from a practical view) is non-discrete, and players don’t have to wait until their turn, hence Quake belongs to the non-discrete game-state class.

In the singleplayer game class, the games only have one simultaneous (human) player, as in case of Solitaire where the human plays, and the computer only shuffles the cards.

The difference between the multiplayer and the massive multiplayer classes is not absolute, but the prior covers games with 2 – 100 simultaneous players (or same order of magnitude), and the latter covers games with the number of players being \( \gg 100 \). An example of a (non-discrete gamestate) massive multiplayer game is NC Soft’s Lineage with approximately 100,000 simultaneous players [4]. We were not able to find examples of discrete gamestate massive multiplayer games.

Another classification schema for computer games is based on genres (e.g. action games, role-playing games, adventure games, sports game etc.) [6]. Genres are suited to classify what the game is about, but not so well suited to classify the amount and what type of data that can be gathered from the game, hence being less useful in a data mining context.

Games can also be classified according to their network architecture (e.g. single node, peer-to-peer, client/server or server-network) [8], which is useful for describing where to collect the data, but doesn’t say anything about what type of data to collect.
3 Game Usage Mining Concept

In figure 2 a mobile massive multiplayer game setting is shown. Players interact with the Massive Multiplayer Game Service (MMGS). The MMGS sends events about user actions to the Game Usage Miner (GUM), if the GUM discovers patterns of interest (e.g. player logoff probability) they get passed on to Recommender Service (RS). When the RS receives a pattern it will give a recommendation back to the MMGS about how to alter the player experience. The purpose of this is to improve the gaming experience for players. Methods used in the RS can e.g. be collaborative filtering or case-based reasoning. Another purpose of the GUM is to provide real-time metrics of the game, typically macro numbers such as mood of the community (e.g. number of players likely to log off in the near future).

![Diagram of Game Usage Mining Concept](image)

**Figure 2: Game Usage Mining Concept**

4 Web vs Game Usage Mining

In this section we compare information gathering in non-discrete game-state massive multiplayer games with information gathering from web browsing. The purpose of the information gathering is to enable game usage mining and web usage mining, respectively.

For web usage mining, the primary information source is the highly standardized web log (i.e. common or extended log format) created by the web server. Unfortunately, there are currently no standardized information sources that supports game usage mining in massive multiplayer games.

4.1 User and Player Session

User session time $t_{session}$ is an important concept in web usage mining, it is the estimate of user’s active period, i.e. before browsing to another web site or stop surfing.

In game usage mining we propose an approach for estimating player sessions. However, if the player pays per time unit to play, it is usually very simple to determine $t_{session}$, since the user probably will explicitly log off instead of idling, or a least be logged out automatically after a particular idle time of the client (in order to avoid high bills). If the client tells the game server that the logout was caused by timeout for a period $t_{idle}$, the accurate session time $t_{session}$ can easily be determined by:

$$t_{session} = t_{stop} - t_{start} - t_{idle}$$ (1)

Where $t_{start}$ and $t_{stop}$ are timestamps for the start and stop of player session, respectively.

4.2 Log rate

Logging of usage data in web usage mining is purely event driven. When the client (browser) requests a web page or an object it causes a log entry line ($LogEntry_k$) to be written to the web server log. The time resolution of the log is usually limited to whole seconds, so if the web server receives several requests in a second, they are written to the web log in an arbitrary order.

Game playing generate many more events than web browsing, since events are based on real-time actions in the game (e.g driving a car,
talking to other avatars, flying on a dragon’s back etc.). To avoid overload of the game service’s logging mechanism, it has to collect events in memory for a period $t_{collect}$, then possibly prune events of less importance and write the important events during the period $t_{collect}$ to the log. If $t_{collect}$ is relatively short (e.g., 1s or less, depending on game type), we propose that game log entries (from possibly several players) collected in that period can have an arbitrary order as the case for web logs. The reason for this is to avoid costly synchronization in parallel implementations.

4.3 Player Information Attributes

This section describes information attributes of the player – the human customer of the gaming service.

Player identification

The user’s IP address is the simplest identification mechanism in web usage mining. However, since many users can have the same IP address (e.g., if surfing through a firewall or proxy gateway), the IP address may not provide a unique user identification.

User identification can be improved by using cookies (small information files residing at the user’s browser which can be written to and read by the web server), but cookies have gotten a lot of critics due to privacy issues. This has resulted in that many users disable the cookie support in their browsers.

Player identification ($player_{id}$) is in games is usually much simpler, since the player is becoming uniquely identified when entering the game (e.g., by username or by mobile phone number) and stays identified the whole playing session.

Player Position

For wireless games, the position $player_{pos}$ of the (human) player may be possible to detect (e.g., using GPS positions obtained from the player’s wireless device). With the $player_{pos}$, interesting attributes such as $player_{speed}$ and $player_{direction}$ can be estimated (during a session), this can possibly add useful knowledge in a game usage mining context.

Player Characteristics

On the web it is possible to get user system characteristics such as browser version and operating system from the extended web log format.

Games on the other hand, can possibly obtain more interesting data about users, information might include the player’s real-life name, address, e-mail, connecting IP address etc (based on manual input by the player when registering).

4.4 Avatar Information Attributes

This section describes information attributes of the avatar – the virtual character that the player controls in the game.

Avatar Actions

A web browsing user generates actions in form of HTTP requests, the most frequent request-type is “GET” used to retrieve pages or objects, but also “POST” and “HEAD” occur relatively frequently.

Action types ($avatar_{action}$) in games are usually much more plentiful and represented as verbs, examples include “jump”, “fire”, “walk” and “say” etc.

Parameters of Avatar Actions

Parameters of an action in web browsing are usually an encoded path requesting a particular HTML file, application/script or multimedia object. This path may contain its own parameters, e.g., if the path requests a script that sets some variables.

Examples of action parameters ($avatar_{actionparameters}$) in games include: natural language or slang phrases (corresponding to action “say”), power or distance (corresponding to action “fire”).
Avatar Path

On the web users can browse between web pages, objects and scripts (discrete movement) by clicking on URLs. However, if items are cached (e.g. by a reverse or standard proxy server, a content provisioning service or the browser cache), browsing will not be detected and logged by the web server. To improve the quality of web logs, java-applets that send messages from the browser to the server can be applied; other attempts to improve path quality include various heuristic methods [3, 9].

In games it is easier to determine the complete path, but the path is more complex to represent than on the web since it’s not necessarily discrete. The player can move the avatar in any direction and not only select between a relatively small set of directions. Possible approaches to represent paths include storing the global game coordinates of the avatar’s position (\( \text{avatar}_{\text{pos}} \)) at every \( t_{\text{collect}} \) period, or game area relative coordinates (\( \text{avatar}_{\text{pos,area}} \)), generalized direction (\( \text{avatar}_{\text{dir}} \)) and speed (\( \text{avatar}_{\text{speed}} \)) together with position data.

Avatar Characteristics

Avatar characteristics may include gender, type of avatar (e.g. elf, troll), occupation (e.g. wizard, warrior) etc. This information can be used in creating profiles of the players.

5 Common Game Log Content

This section summarizes the presented usage attributes found in games, and proposes content for a common game log.

To reduce duplicate information, the game log is proposed divided into two files: 1) \( \text{player.log} \) - which contains information about the user that doesn’t need to be repeated for each action the player does. This file is updated only for every new session, it’s maximum update frequency is \( \frac{1}{t_{\text{session}}} \); and 2) \( \text{playeractions.log} \) - which contains information logs of the players action, it’s maximum update frequency is \( \frac{1}{t_{\text{collect}}} \).

Attributes of \( \text{player.log} \)

\( \text{player_id}, \text{session_id}, \text{timestamp}, \text{player_characteristics}, \text{avatar_characteristics} \)

Attributes of \( \text{playeractions.log} \)

\( \text{player_id}, \text{session_id}, \text{player_pos}, \text{timestamp}, \text{avatar_pos}, \text{avatar_action}, \text{avatar_action parameter} \)

5.1 Log file format

One problem with usage log files in massive multiplayer games with thousands of players is that they’re likely to grow (in size) very rapidly, this has to be considered when storing the data.

An intuitive way of representing the data would be to define XML Schemas or DTDs for \( \text{player.log} \) and \( \text{playeractions.log} \). Unfortunately XML is quite verbose, so an easily compressed, though open and standardized, file format is probably preferable over XML. However, the best approach is probably to allow several types of game log encoding, e.g. both XML and a compressed format. The requirements for a multi-format solution should be simple translation mechanisms between the formats.

5.2 Game Usage Mining Process

Figure 3 shows the details the Game Usage Mining part of figure 2.

Information gathering of game log files described earlier is shown in figure 3, part A. In order the make the system scale up to a large number of players, we need to create higher-level aggregated information, e.g. a sequence of rapid avatar position changes will be interpreted as running. This aggregation process is shown in figure 3, part B.
6 Conclusion

The contribution of this paper has been threefold, first we defined types of data mining in computer, second we provided a classification of computer games from a data mining viewpoint, and third we compared information gathering in web usage mining and game usage mining as well as proposing a common game log format to enable game usage mining.

Future work include determining the actual representation of game log files (e.g. which attributes and which concrete efficient representation), determine what type of usage mining is most useful in massive multiplayer computer games (e.g clustering, classification and incremental sequence mining [10, 7]), and how existing web usage mining architectures and systems can be adapted to a support scalable game usage mining setting.

References


