Grandmaster Story

Nick Feeney, Mike Buerli, Eric Buckthal, and Connor Lange
Computer Science Department
California Polytechnic State University
San Luis Obispo, CA
{ndfeeney,rclange,mbuerli,ebuckthal}@calpoly.edu

***Abstract*—In this paper we explore the possibility of using chess games to generate stories. We built a chess game parsing system that could read in and track chess games from the standard PGN format. Next, we defined eleven features which we believed described the state of the game over a set of moves. Using these features we attempted to generate stories based on the chess game through the use of handcrafted story skins. To accomplish the story generation we build a plot tree iterator that was able to handle, text replacement, plot tree iteration, and resources management. We were able to generate fairly good stories that utilized the chess game to determine the plot of the story. Depending on the chess game different events would happen in the plot. Unfortunately, we found that to have the plot of the story closely match the events of the chess game the input story skin needed to be very detailed with many plot nodes. As of now a detailed story skin requires a large amount of human input.**

***Index Terms*—story generation, drama analysis**

I. Introduction

This paper was written to explore the possibility of utilizing chess games to automatically generate stories. We theorised that events that transpire during a chess game could be translated into plot points which could be then combined to create a story. Our solution is comprised of five main sections, chess game parsing, chess game state reconstruction, chess move feature extraction, story plot iterator, and finally story skins. First, we built a PGN chess game parsing system that we used to parse meta data and chess moves from chess games we obtained from the internet. Next, we built a system that could keep track of the current state of the chess board after every chess move. We needed this system because the PGN file format does not include information about the current state of the game it simply lists the moves made during the chess game. We then created a list of eleven features that could be pulled out of a series of chess moves. These features ranged from how dramatic a move was to how far a piece traveled in a set of moves. Next, we built a story plot iterator that assigned chess moves to story plot nodes and then generated text that was bound together to form our stories. To generate the text for a plot node we created a story skin system. These skins were responsible for the text generation. Each skin was capable of utilized text replacement as well as resource management to generate text blocks. In the solution section we go into more detail about how each piece works, for an overall system design see Figure 1.

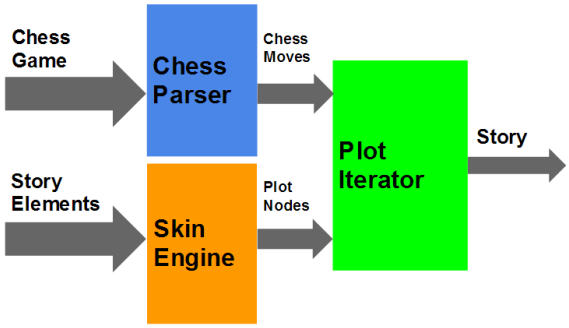

Fig. 1. The Grand Master Story

A. Paper Layout

The structure of the rest of the paper is as follows: Section 2 describes previous work, Section 3 explains the details of the Grand Master Story, Section 4 analyzes the results of the Grand Master Story, Section 5 outlines future work and Section 6 concludes.

II. Related Work

A. Story Generation

B. Chess Analysis

III. Grand Master Story Design and Implementation

A. Chess Game Extraction

To generate stories from chess games, we first needed to obtain a large number of chess games to use as our data set. We gathered 2,367 games from __ in the form of Portable Game Notation (PGN) files. Since each game was in a known format, we simply extracted the information from each game using a regular expression and stored it in a Python dictionary. Each game included the following custom tags, in addition to the standard PGN set [1] and the set of moves in the game:

- WhiteELO - the ranking of the white player (not always present)
- BlackELO - the ranking of the black player (not always present)
- WhiteCountry - the home country of the white player
- BlackCountry - the home country of the black player
- PlyCount - the total number of half-moves played

B. Chess Game State Reconstruction

The chess moves __ from the chess parser, represent a set state diagram of the chess game. The only information that is given is the type of piece and destination, as well as if there was a capture or some other activity. Iterating through this list of moves, we are able to disambiguate the correct chess piece as well as the starting location of the piece being moved. In this way we can construct a list of chess moves that not only keeps track of the current state of the game and the chess board, but provide features of each individual move. Moves can be further analyzed by calculating features of the game, like the total number of threatened pieces as well as the number of targeted pieces (pieces that can be captured in the next turn). Ultimately the set of chess moves will be grouped by the plot iterator and features will be extracted from groups of moves.

C. Chess Move Feature Extraction

The features that were developed for this project attempted get a general overview of the effect that the group of chess moves had on the game. The exact effect of a feature on the chess game is entirely dependent on the skin that is provided for the story. The exact way a chess game feature is use is explained in the Plot Iterator section. In the end, there were eleven features, Dramatic, Danger, Hero, Travel, Unimportant Death, Important Death, Unimportant Kill, Important Kill, Check, Safety, and Defeat.

- Dramatic was a feature that we used to measure how many pieces that a player had in threat.
- Danger was a feature that we developed to measure how many pieces that a player owns and are in threat.
- Hero was a feature that we developed to signal when a single piece takes or kills two or more pieces in a set of moves.
- Travel was a feature that we used to signal when a large amount of the chessboard was covered by the pieces of a player.
- Unimportant Death signaled when a pawn was lost.
- Important Death signaled when a none pawn piece was lost.
- Unimportant Kill signaled when a pawn was killed.
- Important Kill signaled when none pawn piece was killed.
- Check is the feature that signals when a player puts the other players king into check.
- Safety was the feature signalling that a player won the game.
- Defeat signals when a player lost a game.

D. Skins

1) Plot tree: A plot tree is defined as a set of plot nodes, or happenings, in a given story. In a given skin, the plot is constructed by a list of plot nodes that each contain a list of indices, describing the set of next possible plot nodes. In this way we can map plot happenings, restricting linear portions of a storys plot, while also modeling multiple plot branches. This architecture also allows for plot nodes to link back to previous nodes. Plot Node Template/Wordset Story Skins

E. Plot Iterator

The Plot Iterator is the component of Grand Master Story that generates story content from chess move features and a plot graph. Each portion of the iterator is discussed in detail in the following subsections.

1) Chess Move Grouping: Since most of the games we extracted tended to have upwards of 50 moves in them, we couldnt map plot nodes (of which there are less than 50) to each individual move in a game. We originally attempted to use aggregate statistics about the game , in conjunction with examining the features of a particular move to determine importance, to influence a weighted drop rate when pruning moves. However, this lead to less than satisfactory results and we ultimately decided to utilize a uniform grouping of moves. The features for each individual move in the grouping are summed to produce the features of the move grouping.

2) Plot Tree Traversal: The traversal of the plot graph starts with the first plot node, which is fixed for a given skin. The plot iterator then traverses each node one-by-one, calling each nodes generateText() method to build parts of the story, until it gets to a potential branch (either an omission of an event or a different plot path) in the plot graph. To determine which path to take in the plot graph, the iterator calculates the total feature correlation of the move grouping by summing the weights of each feature for each move. Whichever plot node in the graph has the most similar value to the calculated feature weights of the move grouping is chosen. The exact method of calculating feature weights is discussed in the following section. The plot iterator then repeats this process until it gets to the end of the plot graph.

3) Finding the Best Marched Feature: We tried a few different matching algorithms when attempting to match the chess game features with the story skin features. In the end we got the best results with a very simple direct matching method. Our algorithm compared the features of the chess game with the features of the story plot node and found how many overlapped. We then took to overlapping score and subtracted the difference in the total number of features. This method simply favored plot nodes that had the most features in common and had a similar amount of features.

IV. Results

V. Future Work

This project has many pieces that could be expanded for future work. The most obvious and probably the most worthwhile would be to automate the story skin generation. The main limiting factor of this entire project is that the story skins that are needed to generate new stories are handwritten. If there was a way to automatically generate these skins then better and more varying stories could be generated which would also require much less human input.

Another main area for expansion on this project would be feature extraction. We developed a small but potent feature set. Smaller features or less obvious features could be developed which could add more detail into the stories. Character tracking could also be implemented and provide a plot that matched the chess game more accurately. Character tracking

would mean that each chess piece is assigned to a character in the story and the fate of that chess piece would dictate that characters actions.

VI. Conclusion

This project was performed to test the validity of generating stories based on the events that transpire in a chess game. Our implementation proves that it is in fact possible but there is still much room for improvement. The chess game parser does a good job parsing the PGN file format. The chess game tracker does a really good job tracking the state of the game. The chess game feature extraction does a good job getting the central themes of the game but could be expanded to get more detailed features. The plot iterator does an excellent job managing the story skins. The current version of the story skins do a good job at generating text but require large amounts of user input to create. As it is now each story skin takes about about two hours to write. We believe that with a few modifications a similar system will perform extremely well and require much less human input. This topic shows a great deal of potential and hopefully will be expanded in the future.

VII. Acknowledgment

References

[1] Wikipedia, "Portable game notation," http://en.wikipedia.org/wiki/Portable_Game_Notation, 2012.