A Case Base Planning Approach for Dialogue Generation in Digital Movie Design

Sanjeet Hajarnis, Christina Leber, Hua Ai,
Mark Riedl, and Ashwin Ram

College of Computing, Georgia Institute of Technology,
Atlanta, Georgia, U.S.
{sanjeet,cleber3}@gatech.edu,
{hua.ai,riedl,ashwin}@cc.gatech.edu

Abstract. We apply case based reasoning techniques to build an intelligent authoring tool that can assist nontechnical users with authoring their own digital movies. In this paper, we focus on generating dialogue lines between two characters in a movie story. We use Darmok2, a case based planner, extended with a hierarchical plan adaptation module to generate movie characters' dialogue acts with regard to their emotion changes. Then, we use an information state update approach to generate the actual content of each dialogue utterance. Our preliminary study shows that the extended planner can generate coherent dialogue lines which are consistent with user designed movie stories using a small case base authored by novice users. A preliminary user study shows that users like the overall quality of our system generated movie dialogue lines.

Keywords: case based planning, story generation, dialogue generation.

1 Introduction

Recent advances in artificial intelligence (AI) techniques radically change the way users interact with computer entertainment systems. Users are no longer passive consumers of built content, but have become involved in adding value to computer entertainment systems by providing their own content. User generated AI has been deployed in computer games to integrate user-designed strategies into real-time strategy games [17, 28], as well as to train virtual avatars to play in user-defined styles in virtual reality games [14]. Although game designers define the games, user generated game behaviors can tailor character specificities while taking advantage of professionally crafted game space. Moreover, a new trend in the computer entertainment industry is to assist users in creating entertaining contents themselves. Users can now create their own plots in games using intelligent authoring tools [7]. Furthermore, intelligent systems with AI techniques are designed to assist users to build highly dynamic game plots [13]. It is generally believed that users are more engaged in games partially or fully created by themselves or their peers.

Fig. 1. A screen shot of a movie scene generated by Cambot

Besides generating game plots, we are also witnessing new interests in user-generated digital media, which serves as the sole content source for popular social websites such as YouTube^{TM}, Flickr^{TM}, and many others. YouTube recently launched a new portal where users can design their own video clips using animation tools such as GoAnimate^{TM} or Xtranormal^{TM} to build custom videos featuring their own story plots, dialogues, and virtual avatars[1].

While such tools make it possible for non-technical users to design their own media contents, it is still hard for novice users to manage story writing and animation design at the same time. In this paper, we describe our recent effort in using AI techniques, i.e., planning and conversation generation, to build an intelligent system for non-technical users to design their own digital media contents in the format of animated digital movies. We identify two key challenges in this task. One is to design a compelling and interesting story plot, and the other is to render animation. We use Cambot [18] (see Fig. 1.), a virtual movie director for 3D worlds to shoot and edit animation scenes into a movie. In this paper, we describe our work on applying case based planning to assist users in generating story plots that are expressed through dialogue lines between two movie characters. More specifically, we have designed and implemented a case based planning system to fill in gaps in user-generated story contents. We assume that when a user wants to design her own movie, she has a general idea about what story she wants to deliver, e.g., several key movie scenes. However, the user may find it tedious to author the full details in the story or alternatively, novice users would find it difficult to author stories from scratch. Our system is designed as an intelligent authoring tool, which can automatically fill in story contents to free users from the authoring burden, or suggest story content so that users can revise and edit them as per their will.

We consider applying Case Based Planning techniques in story generation because such techniques can utilize interesting story contents generated by previous users to enrich new users' stories. Case Based Planning has been applied to a variety of tasks, including computer game AI design, robotics and story generation [6, 9, 22]. It is planning as remembering [8], which involves reusing previous plans and adapting them to suit new situations. However, applying case based reasoning for story

[1] Can be created at YouTube.com/create.

generation is different from the well-studied applications in the game domain [9]. In a game, there is a finite set of actions that a user or an avatar can take. But in story generation, there is not a defined set of actions that can happen in the story. Instead, the characters in the story can perform any actions that are suitable given a certain story context. There have been query based approaches to restrict the content for using Case Based Reasoning in story generation [6]. Another case based reasoning solution is used for story representation and adaptation in the fairy tale domain where the CBR uses cases extracted from a multi-move story scripts given by Propp [19]. In this study, we code character behaviors at an abstract level in order to utilize similarities among different action sequences in case based planning. We define character behaviors in regards to the behaviors' effects on characters' emotion changes because we believe emotion expression is an important factor in movie story generation.

The rest of the paper is organized as follows. Section 2 surveys different applications of case based planning as well as story and dialogue generation. Section 3 defines our task domain. Section 4 introduces our case based planning system. Section 5 describes a preliminary evaluation. We conclude in Section 6 and point out future directions.

2 Related Work

Case-based reasoning (CBR) is a known problem solving technique based on reutilizing specific knowledge of previously experienced problems stored as cases [1], [11]. The CBR cycle consists of four major stages: Retrieve, Reuse, Revise and Retain [1] (See Fig. 2). In the Retrieve stage, the system selects a subset of cases from the case base that are relevant to the current problem. The Reuse stage adapts the solution of the cases selected in the retrieve stage to the current problem. In the Revise stage, the obtained solution is verified (either by testing it in the real world or by examination by an expert), which provides feedback about the correctness of the predicted solution. Finally, in the Retain stage, the system decides whether or not to store the new solved case into the case base [16].

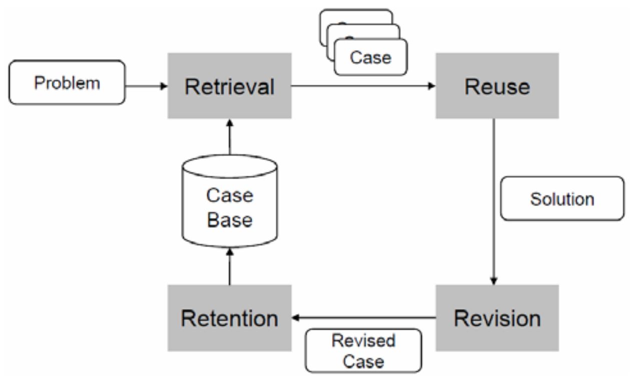


Fig. 2. The CBR cycle

We further explore varied case-base reasoning and case-base planning applications based on different underlying domains and purposes. Case-base reasoning and planning are generally used when generating plans from scratch can be computationally expensive or when many new problems can be solved using solutions from previous experiences. There have been numerous such case-based planning techniques that adhere to the above philosophy [9].

Prodigy/Analogy [27] is such an architecture which stores the reasoning trace while planning instead of storing the plans. Thus, when planning for a new problem, the system just replays the stored decision traces. While Prodigy/Analogy is an example of the derivational plan adaptation approach, CHEF is an example of the transformational plan adaptation approach [15]. PRIAR is yet another planner system that focuses on reusing previously annotated plans with a validation structure that contains an explanation [10]. This system requires annotated plans to ensure appropriate reusing of the stored plans. The MAYOR [5] system contains a concept net of all the factors in the game domain and how they are related to each other. These factors include money, crime, pollution etc. Each plan in the plan library has some expectations on the aforementioned factors. While planning, if the expectations on the factors are not complete, the concept net allows the system to manipulate the expectation failure. The above CBR systems are domain dependent and function on different domains from dialogues or conversations. HICAP [2] is a general-purpose planner that integrates hierarchical plan base structure with constraint satisfaction based on domain-dependent doctrines.

Story Generation can be viewed as a planning problem. It is a problem of crafting a structured sequence of events that can be told to a recipient [23]. A particular planning problem can be thought of as follows: Given an initial state, an underlying domain and a goal state, find a sequence of operations that transform the world from the initial state to the goal state where the operators adhere to the requirements in the domain. Such a planning problem can be solved by different types of planners. One particular class of planners that can be used to solve the above problem is called partial-order planners (POP) [20]. A partially-ordered plan consists of a set of actions that are ordered according to a set of temporal constraints forming a partial ordering such that some actions may be unordered relative to each other. Moreover, there exists causal links between two actions which imply that the first action creates changes in the world that enable the second action to be performed. POP planners encapsulate many features that appear in the cognitive models of narrative [29] and hence prove useful in narrative generation. As we mentioned in Section 1, case based planning systems have also been used in story generation [6]. Moreover, there have been attempts to combine case-base reasoning and planning algorithms to develop a better system. MEXICA uses elements of both previous story retrieval and means-ends planning [21]. Here, we perform a similar task by extending a case based planning system with hierarchical planning strategies.

In this study, we handle a subtask of story generation, which is to generate dialogue lines for the characters in a narrative story. This is a very important part in generating stories for movies. Research on dialogue generation has been mainly on information providing dialogue systems [24, 25], which are used to provide information to users in order to complete user defined tasks, such as booking flight tickets, or querying weather conditions. In these task-oriented dialogues, dialogue contents can be represented by a set of information slots. For example, in a flight booking conversation, the dialogue system needs to know information about departure and arrival cities to recommend flight options. Dialogue generation in this case is centered on updating states of these two basic information slots. The information state update approach is a simple way to keep track of conversation history [26]. We use a similar approach in managing dialogue content in our system. However, this simple approach alone is not enough to generate interesting and entertaining dialogues required in our task. Therefore, we deploy a case based planning system that decides how the dialogue content will be delivered with different emotions. More details are described in Section 4.

3 Task Domain

Our ultimate goal is to build an intelligent system which can help users generate narrative movie stories by filling in gaps between user generated contents. In this study, we focus on generating dialogues between two movie characters, because dialogues are an important way to create engaging experience in movies. Dialogue can not only deliver movie plots, but also represent the personalities of movie characters and their emotions.

In this study, our task is to generate dialogues between two main characters namely Great Goblin and Thorin, given the background story:

> *When crossing the Misty Mountains, Thorin and his company run in to a storm and take shelter in a cave. The cave actually serves as an entrance to the lair of the Goblins of the Misty Mountains who sneak into the cave while the company is sleeping and capture them. The captives are brought before the Great Goblin, who queries about why they were in his cave. Thorin speaks for his party and tries to negotiate with the Great Goblin and get released.*

When a user specifies the emotion states for the two characters, our system can generate a dialogue based on character emotions and the background story. We currently support 3 emotion states: calm, fear and anger.

4 Case-Based Reasoning for Dialogue Generation

We choose to use a case-based reasoning approach for dialogue generation because we want to make use of user generated dialogues. However, since user generated dialogues may happen in different background contexts, we cannot directly store these dialogues in the case base and retrieve them for another story. Therefore, we represent the dialogues in their abstract format using dialogue acts (Section 4.1). Then we generate dialogues in two steps. First, we use a case-based planner to generate a dialogue act sequence (4.2). Then, we apply an information state update approach for content generation (4.3). Finally, we use a simple natural language generator to deliver the full dialogue in natural language (4.4).

4.1 Dialogue Act Representation

Dialogue acts are generally used to model and automatically detect discourse structure in dialogue systems. A dialogue act represents the meaning of an utterance at the level of illocutionary force [3]. Thus, a dialogue act is approximately the equivalent of the actions in computer games. Dialogue acts are usually defined to be relevant to a particular application, although there have been efforts to develop a domain-independent dialogue act labeling systems [4].

In our task, we define a set of dialogue acts for a negotiation scene. Table 1 shows the dialogue acts and their definitions. We can see that some dialogue acts are likely to trigger the change of emotions. For example, when one character decline to answer a question, the asker is likely be angry. Similarly, threatening is likely to trigger fear. In Section 4.2, we use a case based reasoning system to generate traces of dialogue acts based on user defined character emotion states.

Table 1. List of Dialogue Acts used by the Natural Language Generation module

Dialogue Acts	Definitions
Question	Ask a question
Answer	Directly answer a question
Dodge	Give an indirect answer to a question
Decline	Refuse to answer a question
Follow-up	Ask a question based on the previous answer/question
Validate	Ask a question to further verify the previous answer
Propose	Propose a solution
Agree	Agree to a proposal
Reject	Reject a proposal
Counter	Follow-up on a rejection with a new proposal
Inform	Provide new information
Provoke	Provide new information intended to anger the listener
Persuade	Statement intended to persuade the listener of something
Threaten	Issue a threat
Warn	Issue a warning
Decide	End of conversation

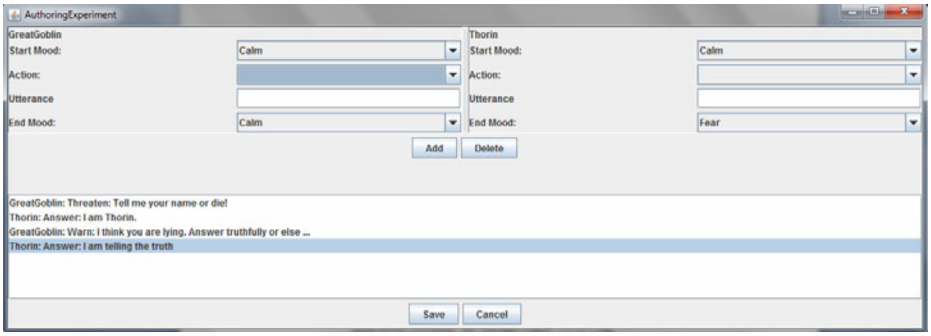
4.2 Case Based Reasoning System

We use Darmok2 [16] for case-based planning to generate dialogue act traces. In our system, since the case base consists of different plans learned from user generated cases, we also refer to it as the plan base. A plan consists of a start state of the characters in the dialogue, a sequence of dialogue actions that occurred between the two characters and an end state of the characters. A state in this dialogue generation domain consists of the characters involved in the dialogue and their emotional conditions. Currently, the domain supports three main emotional conditions namely Calm, Angry, and Fear, but the system is generic to incorporate more emotional conditions. An action comes from the list of 15 actions in the domain.

In order to use Darmok2, we need to provide the system with a case base to support the system's planning mechanisms. In Section 4.2.1, we show how cases are authored and represented in Darmok2. In Section 4.2.2, we describe how Darmok2 can be used to learn plan cases from the traces authored using our authoring interface. In Section 4.2.3, we describe how Darmok2 uses these cases for planning.

4.2.1 Case Base Authoring

The system acts as an intelligent tool to assist story authors. An author can choose to use the system for intelligent recommendations or author the entire story from scratch. When asking for recommendations, the case based planning system is called to fill in story scenes based on the current story. However, before the case based planning system can function, we need to build a case base to support its planning. Therefore, when users are authoring their own stories, story traces are stored in the system and represented as cases to be used in planning later. Fig. 3. shows the story authoring interface.



GreatGoblin		Thorin	
Start Mood:	Calm	Start Mood:	Calm
Action:		Action:	
Utterance		Utterance	
End Mood:	Calm	End Mood:	Fear

```
GreatGoblin: Threaten: Tell me your name or die!
Thorin: Answer: I am Thorin.
GreatGoblin: Warn: I think you are lying. Answer truthfully or else ...
Thorin: Answer: I am telling the truth
```

Fig. 3. Authoring interface for generating stories

The top left of the interface contains Great Goblin's state information (which includes initial mood, dialogue actions and utterances corresponding to the dialogue act and final mood). Similarly, the top right Section contains information about Thorin's state information. The bottom part contains a log of the dialogues so far, which includes both dialogue acts and the actual utterances between Great Goblin and Thorin. To add new dialogues, a user must pick a dialogue action from the list in Table 1. Then, the author can enter the dialogue in the utterance field. Once an action is picked and the utterance written, author can press the add button to include the newly created dialogue in the log.

4.2.2 Case Base Learning

Once stories are generated after the authoring phase, we can enrich our plan base by learning from these traces. Learning from a given trace involves three major steps: (i) detecting goals (ii) building plans (iii) updating the plan base. The learning algorithm detects changes in the consequent states based on the changes in the character's behavior or emotional condition. Once the changes are detected, the learning algorithm must build appropriate plans. There are two main types of plans in the plan base: (i) Goal plan – a plan that satisfies a particular emotional goal. A goal plan is created by directly calling the planner in Darmok2; (ii) Combined plan – a plan which makes use of the hierarchical goal structure to make new plans from goal plans. A combined plan is created by first decomposing the final goal state into several intermediate goal states which can eventually lead to the final state, and then calling the Darmok2 planning on each intermediate goal states. After building appropriate plans from the changes, the learning algorithm enriches the plan base with the newly created plans.

Consider a simple example:

State:
Great Goblin: Calm
Thorin: Calm
Dialogue Acts:
Great Goblin [Threaten]: Tell me your name or die!
Thorin [Answer]: I am Thorin.
State:
Great Goblin: Calm
Thorin: Fear
Dialogue Acts:
Great Goblin [Question]: Are you sure? I think you are lying.
Thorin [Answer]: I am not.
State:
Great Goblin: Angry
Thorin: Fear

The above trace generates the plan structure shown in Fig. 4.

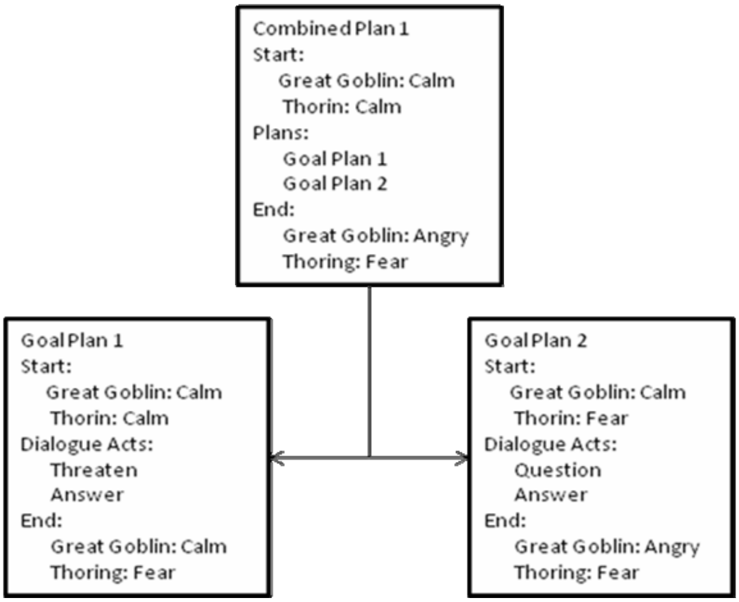


Fig. 4. A sample plan structure

Note that the size of a goal plan is not restricted to only two dialogue acts as shown in the example in Fig. 4. Similarly, a combination plan can have more than two goal plans encapsulated within it. We use the above learning procedure to improve our plan base with both goal plans and combination plans.

4.2.3 Case Based Planning

When an author needs suggestions from the system to fill certain parts of a dialogue, Darmok2 is called to retrieve reasonable plans. The plan retrieval phase ranks the plans in the plan base based on the current state in the dialogue and the target state. Two scores are calculated to retrieve the best plan: (i) goal match score – this heuristic calculates the similarity between the goal state of a plan in the case base and the targeted goal state; (ii) initial state match score – this heuristic calculates the similarity between the initial state of a plan in the case base and the retrieval query's initial state. In our current system, the retrieval phase returns up to 10 different plans and provides the Plan Adaptation phase with more options to choose the right plan.

The Plan Adaptation uses the plans obtained in the retrieval phase to perform dialogue generation. This scenario can be visualized as a gap-filling problem because a typical plan retrieval query would have a start state and end state and the system attempts to generate dialogue for the start and end states. Thus, the system attempts to come up with the most appropriate dialogue to narrate the gap between the start and the end state. The Plan Adaptation module picks the best plan from the list of retrieved plans to fit this gap. There are 4 possible events that could occur in this case: (i) the retrieved plan would completely fit the gap (ii) the retrieved plan would be a complete fit for the start state but not for the end state (iii) the retrieved plan would be a complete fit for the end state but not the start state (iv) the retrieved plan would not fit either the start or the end state. If the gap is completely filled, then the adaptation phase does not have to perform further checking. In events 2-4, the plan adaptation tries to invoke a planner to find the missing pieces. The planner in the plan adaptation phase recognizes gaps (like in events 2-4) and recursively calls the retrieval phase until all the gaps are filled satisfactorily. Thus, the planner in plan adaptation can be thought of as an example of recursive case-base planning. Consider the query with the following initial and final states to the plan-base:

Initial	**Final**
Great Goblin: Calm	**Great Goblin:** Angry
Thorin: Angry	**Thorin:** Fear

Now, in adaptation phase if the best plan so far is from

Initial	**Final**
Great Goblin: Angry	**Great Goblin:** Angry
Thorin: Angry	**Thorin:** Fear

then the planner recognizes the gap and recursively queries the plan-base with

Initial	**Final**
Great Goblin: Calm	**Great Goblin:** Angry
Thorin: Angry	**Thorin:** Angry

such that the gap in the beginning of the retrieved plan is filled satisfactorily.

Moreover the adaptation phase also looks for substitutable plans if the recursive plan adaptation fails. In our current domain, we define approximation rules for emotional changes. For instance, for Thorin's character, an emotional change from Calm to Angry can be approximated by emotional change from Angry to Fear, assuming unpleasant information or actions can make a character's emotion becomes worse. In our story domain, being angry is a worse emotional state than being calm because it shows the character feels unhappy and dislikes the current situation. Being "fear" is worse than being angry because the character feels the threats and may start to react instead of only expressing the emotion. By adding approximation rules, the plan adaptation module ensures that the retrieval query gap gets filled completely. For our application, it is more important for our system to respond to every user query than to always provide a response of good logical sense. Since our system is used to generate movie dialogues, using approximation rules may not always help the system to generate a sensible dialogue, but can possibly produce interesting and funny effects that are also favorable in movie dialogue generation.

4.3 Dialogue Content Generation

Once we have the sequence of actions that will make up the story, we must then assign each dialogue action an utterance, which is a single-sentence string that represents what the character will say. As mentioned earlier, the issue here is that we cannot merely have a one-to-one mapping from action to utterance – a set of utterances that matches one sequence of dialogue actions may not match the same dialogue actions in a different sequence. The reason for this is that different sequences of actions will result in different contexts, and since the sequence of dialogue actions is user generated (though aided by this system), we must be able to support any context that is created. Users will not want to use this system if it cannot generate dialogue that makes sense in the story they create.

To solve this problem, we use an information state update approach to track dialogue content. An utterance library is pre-authored with sets of utterances for each possible dialogue act. Each utterance is tagged with a pre-context and a post-context, which are both represented by a set of topics. The pre-context indicates which topics have already been mentioned in the conversation in order for that particular utterance to make sense. The post-context indicates which new topics the utterance introduces to the conversation and occasionally which topics, if any, the utterance can settle. These topics indicate which information has or has not been disclosed in the dialogue. By updating these information states, we keep track of which topics should be in the dialogue content. Settled topics do not need to be discussed further.

Take the following exchange as an example:

> **Great Goblin**: How do I know you are not here to spy on us?
> **Thorin:** We are not spies.

It does not make sense for Thorin to mention that he is not a spy until he is first accused of being a spy. Therefore, Thorin's utterance would have a pre-context of "spies", since he cannot say that utterance until the topic of "spies" is introduced. The

Great Goblin's utterance would have a post-context of "spies" to indicate that once that utterance is selected to be part of the conversation the topic of "spies" has been introduced.

Internally, the dialogue generation system keeps track of which topics are currently active in the context as part of the information states. When given an action, it determines which of the utterances in the library match both the given action and the current information state. If there is more than one possible utterance that matches, the system looks at the last utterance in the conversation to determine which possible utterance most closely matches the current context. This is done by checking which of the possible utterances' pre-contexts most closely matches the post-context of the previous utterance in the conversation. The system then selects that utterance, sets the utterance as the output of the action, updates the information state according to the post-context of the utterance, and returns the updated action.

Emotion is considered to be part of the context. People speak differently when they are angry than when they are calm or afraid, and our utterance library needs to reflect that. So, some utterances have an emotion as part of the pre-context or post-context. When given an action tagged with a desired emotional outcome, in addition to the procedure outlined above, the system will also check the post-context of each utterance to determine if it will produce the desired emotional outcome. Any utterance that will not produce that emotional outcome is ignored, even if it matches the current conversation state.

4.4 Natural Language Dialogue Generation

In the experiments reported in this paper, the dialogue generation module uses canned dialogue lines rather than generating utterances programmatically. We feared that introducing a natural language generation system could negatively bias the results if we ended up with a sequence of actions for which the natural language generation system did not perform well. We have done some work with the NLG system Personage [12], which adjusts how an utterance is generated based on the personality of the speaker, to see if we could possibly apply it in this domain and cause it to generate an utterance differently based on the emotional state of the speaker in addition to their personality. That work, however, is beyond the scope of this paper.

5 Experiments and Results

We perform a two-phase preliminary experiment to evaluate our system. In the first phase, we invite three users (2 males and 1 female) to write stories using our system. These stories are used to construct the plan base for Darmok2. We choose to use user-generated stories instead of expert-authored stories because we want to evaluate our system in a realistic scenario, in which the planning system uses user-generated stories to provide recommendations upon user requests. In addition, if knowledge is engineered by expert authors, it conflates the creative abilities of the system and those of the authors'. Therefore, we do not use expert-authored stories to bias our system evaluation.

The three subjects were given the background story shown in Section 3. They were asked to generate 3 dialogues between the two characters in the story. None of the users have difficulties writing the stories although only one of them is familiar with English fantasy fictions like our background story. The average length of user generated dialogues is 5.67 turns. The average number of emotion changes per dialogue is 2.5 times. In total, the plan base constructed by user-generated dialogues has 53 cases.

These subjects were then asked to each create 2 cases in which they would want to get story recommendations from our system, i.e., to ask the system to generate the dialogue for them. In each case, they need to specify the start and end emotion states of both characters in the dialogue. The subjects were given enough prior background information about the domain story so that they could make an informed choice for the start-end pairs.

Once we obtained all 6 start-end pairs, we generated dialogues for each pair using three different versions of the plan base. The first version of the plan bases contained 33% of the user generated plans (17 plans) for the plan base to start off with, the second version contained 66% (34 plans) of the user generated plans and the third version contained 100% of the user generated plans (53 plans). Each subset of the plan base was randomly chosen. We want to test the impact of the size of the plan base on story quality. In total, we generated 18 (6*3) dialogues. We recruited a different set of judges to evaluate these dialogues because we did not want the judges to be biased when seeing portions of their own dialogues being used in new ways. We also decrease a judge's bias by assigning one dialogue to two judges and use the average ratings by the two judges as the final ratings of that dialogue. Our hypothesis is that by extending the CBR with a hierarchical planning strategy, the quality of generated dialogues would not be impacted by the different sizes of the plan bases.

We recruited 4 judges (3 males and 1 female) to assess the quality of generated dialogues. The judges were assigned 3 stories each. Each judge was also provided with dialogues generated by all the three versions of the plan base so they would be capable of providing a good comparison. We asked the judges to rate the dialogues on four evaluation measures, i.e., logic coherence, interestingness, fitness with background story, and fluency. We also asked the judges to give an overall rating of how well they like a dialogue. Ratings are on a 4-point Likert scale. We did not use a 5-point likert scale because we wanted our judges to clearly indicate whether our system's performance is good (a rating of 3 or 4) or bad (a rating of 1 or 2). We did not want the judges to give neutral answers.

Table 2. Summary of average ratings across all dialogues and all judges

Logical Coherence	Interestingness	Fitness with background	Dialogue Fluency	Overall
3.03 (±0.56)	2.78 (±0.84)	3.14 (±0.68)	3.22 (±0.52)	2.69 (±0.73)

Table 2 summarizes the average ratings across all dialogues and all judges. The numbers in the parenthesis show the standard deviations. The average overall rating indicates that the judges' overall feedback on our system's performance is positive (2.69). It is not surprising to see that dialogue fluency gets a high rating because the natural language generation module uses pre-authored utterances. Logical coherence

(3.03) and fitness with background (3.14) scores show that the performance of our planning module is also positive. The interestingness of the dialogues (2.78) gets the lowest rating among all the measures. Using Pearson's correlation, we observe that there is a strong correlation between the length of the generated dialogues and its interestingness ($R = 0.71$, $p \leq 0.05$). In other words, judges tend to rate longer dialogues to be more interesting. When controlling for dialogue length, each of our four measures strongly correlate with judge's overall ratings, which shows that all our measures are important aspects that judges consider when rating a dialogue. When we use the four measures to build a regression model to predict the overall rating, fluency and interestingness have the highest coefficients, which show that these two measures have the highest weights in impacting the judge's overall ratings. Since our system generates relatively short dialogues (on average 5 turns) and therefore gets a lower interestingness score, the overall dialogue ratings are not very high. Here is an example of generated dialogues (4 turns) for the following conditions. In the beginning of this dialogue, Great Goblin is calm and Thorin is angry and at the end of the dialogue Great Goblin is angry while Thorin is scared.

Great Goblin: Who are you?
Thorin: I'm not answering your questions.
Great Goblin: Answer me truthfully, and I'll consider letting you live.
Thorin: No. I demand you let us go.

When comparing dialogues generated from the three different sizes of plan bases, we do not observe any statistical significance among any of the evaluation measures or the overall scores (p values larger than 0.05). This supports our hypothesis by showing that our system can work with a small plan base generated by naïve users. Although this result is based on our simple domain, we are encouraged by this result because being able to work with a small plan base makes it possible for users to switch our system to a new story domain quickly.

6 Conclusions

In this paper, we report our recent work on using a case based planning system for narrative story and dialogue generation. Our system acts as an intelligent tool to assist users to write their own movie scripts. When users use our system to write their movie lines, their scripts will be stored in the system. Later, when a user lacks the idea to write certain scenes, she can use our system to fill in the gap in the movie. Our system uses user-generated story to construct its plan base, and then uses the plan base to generate new stories to fill in gaps in new stories. Our task of applying a case based planning system on dialogue generation distinguishes from previous applications in the game domain because system actions in one dialogue need to be represented at a higher level in order to be reused in other dialogues. Therefore, in our system, we code each utterance in dialogues with a dialogue act to represent its conversational strategy and its effect on conversational partners' emotions. Also, we build a content planner using an information state update approach in supplement to the case based planner in order to generate the dialogue content.

Our results show that our system can generate logically coherent stories that reflect background contexts. The generated stories get positive user ratings in terms of their overall quality. Moreover, our system can work with a small case base contributed by naïve users. This feature enables our system to be used by users in new domains.

However, our stories have relatively lower ratings on their interestingness, which is found to be directly correlated with dialogue length. In the current system, we do not have a drama management module that shapes the generated dialogues in terms of its interestingness. In the future, we plan to add heuristic rules which can generate more interesting dialogues, but not the most direct dialogues.

Acknowledgements. The authors gratefully acknowledge the support of the U.S. Defense Advanced Research Projects Agency (DARPA) for this effort.

References

1. Aamodt, A., Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. Artificial Intelligence Communications 7(1), 39–59 (1994)
2. Aha, D.W., Breslow, L.A., Munoz-Avila, H.: Conversational Case-Based Reasoning. Applied Intelligence 14, 9–32 (2001)
3. Austin, J.: How To Do Things With Words. Harvard University Press, Cambridge (1962)
4. Core, M., Allen, J.: Coding Dialogues With The DAMSL Annotation Scheme. In: Working Notes of the AAAI Fall Symposium on Communicative actions in Humans and Machines, Cambridge, MA, pp. 28–35 (1997)
5. Fasciano, M.: Everyday-World Plan Use. Technical Report TR-96-07, University of Chicago (1996)
6. Gervas, P., Diaz-Agudo, B., Peinado, F., Hervas, R.: Story Plot Generation Based on CBR. Knowledge Based Systems 18(4-5), 235–242 (2005)
7. Hajarnis, S., Barve, C., Karnik, D., Riedl, M.O.: Supporting End-User Authoring of Alternate Reality Games with Cross-Location Compatibility. In: Proceedings of the 24th Conference of the Florida Artificial Intelligence Research Society, Palm Beach, FL (2011)
8. Hammond, K.F.: Case-Base Planning: Viewing Planning as a Memory Task. Academic Press, New York (1989)
9. Hammond, K.F.: Case-Based Planning: A Framework for Planning from Experience. Cognitive Science 14(3), 385–443 (1990)
10. Kambhampati, S., Hendler, J.A.: A Validation-Structure-Based Theory of Plan Modification and Reuse. Artificial Intelligence 55(2), 193–258 (1992)
11. Kolodner, J.: Case-based reasoning. Morgan Kaufmann, San Francisco (1993)
12. Mairesse, F., Walker, M.: PERSONAGE: Personality Generation for Dialogue. In: Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL), Prague (2007)
13. McCoy, J., Treanor, M., Samuel, B., Tearse, B., Mateas, M., Wardrip-Fruin, N.: Authoring Game-Based Interactive Narrative using Social Games and Comme il Faut. In: Proceedings of the 4th International Conference & Festival of the Electronic Literature Organization: Archive & Innovate (2010)
14. Mehta, M., Ram, A.: Run Time Behavior Adaptation for Real Time Interactive Games. IEEE Transaction of AI and Games 1(3) (2010)

15. Munoz-Avila, H., Cox, M.: Case-Based Plan Adaptation: An Analysis and Review. IEEE Intelligent Systems (2007)
16. Ontañón, S., Sugandh, N., Mishra, K., Ram, A.: On-line Case-Based Planning. Computational Intelligence 26(1), 84–119 (2010)
17. Ontañón, S., Ram, A.: Case-Based Reasoning and User-Generated AI for Real-Time Strategy Games. In: Gonzales-Calero, P., Gomez-Martin, M. (eds.) AI for Games: State of the Practice (2011)
18. O'Neil, B., Riedl, M.O., Nitsche, M.: Towards Intelligent Authoring Tools for Machinima Creation. In: Proceedings CHI, Extended Abstracts, Boston, MA (2009)
19. Peinado, F., Gervas, P., Diaz-Agudo, B.: A Description Logic Ontology for Fairy Tale Generation. In: 4th International Conference on Language Resources and Evaluation: Workshop on Language Resources and Linguistic Creativity (2004)
20. Penberthy, J.S., Weld, D.S.: UCPOP: A Sound, Complete, Partial-Order Planner for ADL. In: Proceedings of the 3rd International Conference on Knowledge Representation and Reasoning, pp. 103–114 (1992)
21. Pérez y Pérez, R., Sharples, M.: MEXICA: A Computer Model of a Cognitive Account of Creative Writing. Experimental and Theoretical Artificial Intelligence 13, 119–139 (2001)
22. Ram, A., Santamaria, J.C.: Continuous Case-Based Reasoning. In: Proceedings of the AAAI 1993 Workshop on Case-Based Reasoning, Washington DC, pp. 86–93 (1993)
23. Riedl, M.O.: Story Planning: Creativity Through Exploration, Retrieval and Analogical Transformation. Minds and Machine 20(4), 589–614 (2010)
24. Rieser, V., Lemon, O.: Learning Effective Multimodal Dialogue Strategies from Wizard-of-Oz data: Bootstrapping and Evalutaion. ACL (2008)
25. Thomson, B., Young, S.: Bayesian Update of Dialogue State: A POMDP framework for spoken dialogue systems. Computer Speech and Language 24(4), 562–588 (2010)
26. Traum, D., Larsson, S.: The Information State Approach to Dialogue Management. In: van Kuppevelt, J., Smith, R. (eds.) Current and New Directions in Discourse and Dialogue, pp. 325–353. Kluwer, Dordrecht (2003)
27. Veloso, M., Carbonell, J., Perez, A., Borrajo, D., Finkan, J., Blythe, E.: Integrating planning and learning: The prodigy architecture. Experimental and Theoretical Artificial Intelligence 7 (1995)
28. Weber, B.G., Mateas, M.: Case-Based Reasoning for Build Order in Real-Time Strategy Games. In: The Artificial Intelligence for Interactive Digitial Entertainment (2009)
29. Young, R.M.: Notes on the use of plan structures in the creation of interactive plot. In: Mateas, M., Sengers, P. (eds.) Narrative Intelligence: Papers from the AAAI Fall Symposium (Technical Report FS-99-01). AAAI Press, Menlo Park (1999)