Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence

Story Generation with Crowdsourced Plot Graphs

Boyang Li, Stephen Lee-Urban, George Johnston, and Mark O. Riedl
School of Interactive Computing, Georgia Institute of Technology
{boyangli; lee-urban; gjohnston3; riedl}@gatech.edu

Abstract

Story generation is the problem of automatically selecting a sequence of events that meet a set of criteria and can be told as a story. Story generation is knowledge-intensive; traditional story generators rely on *a priori* defined domain models about fictional worlds, including characters, places, and actions that can be performed. Manually authoring the domain models is costly and thus not scalable. We present a novel class of story generation system that can generate stories in an unknown domain. Our system (a) automatically learns a domain model by crowdsourcing a corpus of narrative examples and (b) generates stories by sampling from the space defined by the domain model. A large-scale evaluation shows that stories generated by our system for a previously unknown topic are comparable in quality to simple stories authored by untrained humans.

Introduction

The ability to craft, tell, and understand stories has long been held as a hallmark of human intelligence (Winston 2011) and a long-term goal of artificial intelligence. Story generation, storytelling, and story understanding are examples of a phenomenon called *narrative intelligence*, which we define as an entity's ability to (a) organize and explain experiences in narrative terms, (b) comprehend and make inferences about narratives we are told, and (c) produce affective responses such as empathy to narratives.

Prior story generation research built on artificial intelligence techniques such as planning (e.g., Meehan 1976; Riedl and Young 2010) or case-based reasoning (e.g., Turner 1994; Gervás et al. 2005). These prior approaches to story generation rely on an *a priori* known *domain model*—a description of a fictional world, including characters, objects, places, and the actions that entities can perform to change the world. Once the domain model has been engineered, a story generation system can tell a potentially infinite number of stories involving these characters, places, and actions known to the system.

In this paper, we explore a new narrative intelligence challenge: *open story generation*, the ability to autonomously create narratives about any topic without relying on *a priori* engineered domain models. Based on the insight that humans possess a lifetime of experiences to call upon when creating new stories, our approach is to learn the domain model from simple stories supplied by humans in the form of crowdsourcing. We present an open story generator that (1) crowdsources a corpus of narrative examples of a new domain, (2) automatically constructs a domain model, and (3) samples from the space of stories allowed by the domain model according to some story quality criteria.

This paper extends prior work toward learning domain models (via corpus and/or crowdsourcing) that were either geared for narrative understanding to support the requirements of narrative generation. Specifically, learning of mutual exclusion relations and optional events allows coherent stories to be generated from a domain model capturing different possible, non-contradictory story trajectories. We describe our story generation algorithm and report on results of a large-scale evaluation of the stories generated by our system, which indicate performance on par with non-expert human storytellers.

Plot Graphs

A commonly used representation in story generation systems is the *plot graph*, which models the author-intended logical flow of events in the virtual world as a set of precedence constraints between plot events (Weyhrauch 1997). A plot graph defines the space of legal story progression and ultimately determines possible events at any given point in time. For example, a plot event of the player opening a vault must be preceded by plot events in which the player finds the vault and acquires the vault's key (Nelson & Mateas 2005). Plot graphs may contain OR-

nodes denoting events that can become available to be included in the story in different ways.

Our plot graph representation differs slightly from that of previous work. In this paper, a plot graph is a tuple $G = \langle E, P, M \rangle$ where E is the set of plot events, $P \subseteq \{x \rightarrow y | x, y \in E\}$ is a set of ordered pair of events that describe precedence constraints, and $M \subseteq \{(x, y) | x, y \in E\}$ is a set of unordered mutual exclusion relations. Mutual exclusion relations indicate which plot events can never co-occur in the same narrative, performing the same function as OR-relations but are more general.

Related Work

Automated story generation is the problem of automatically selecting a sequence of events or actions that meet a set of criteria and can be told as a story. Typically this is performed at the level of *fabula*—roughly, plot points—where events are conceptual-level constructs. Once a story—a sequence of plot points—is created it can be used to further generate natural language or to control characters in a virtual world. Story planning systems attempt to solve the problem of selecting and sequencing character actions to achieve a set of given success criteria, such as an outcome state or dramatic arc (Meehan 1976; Lebowitz 1987; Riedl and Young 2010; Porteous, Cavazza, & Charles 2010). Case-based reasoning systems adapt prior stories (cases) to new storytelling requirements (Turner 1994, Gervás et al. 2005). Other approaches include agent-based simulation (Brenner 2011).

Most story planning and case-based reasoning story generation systems require a domain model that provides knowledge about characters and places in a fictional world and a set of possible actions that characters can perform in that world. One exception is the SAYANYTHING interactive story system (Swanson & Gordon 2012) that mines lines of text from a corpus of blogs scraped from the Web. Because global context is hard to acquire from natural language texts, computer and human co-creators alternate lines of the story. The MAKEBELIEVE system extracts commonsense rules about action sequences from the Open Mind knowledge base (Liu & Singh 2002). Commonsense knowledge bases typically focus on declarative facts about non-fictional topics limiting their applicability to storytelling. These systems use open knowledge sources: a Web corpus or a crowdsourced knowledge base.

Our approach builds off recent work in story script acquisition from natural language texts. Chambers and Jurafsky (2008) extract events and temporal relations from news corpora for common activities. Their script representation is a set of events with temporal precedence constraints—similar to the plot graph but does not contain OR relations. Regneri, Koller, and Pinkal (2010) acquire scripts from crowdsourced "bullet-list" style instructions entered by anonymous individuals and then manually converted to a canonical form. It uses a script representation similar to that of Chambers and Jurfasky. Neither work attempts to generate stories from the acquired scripts. McIntyre and Lapata (2010) attempt to learn a content planner for stories from a corpus of fairy tales but only achieve local coherence between story events.

The scripts learned by the above techniques do not recognize the fact that it is possible to pick up events that represent distinct alternatives (e.g., driving through a fast food restaurant versus dining in) or events that should never co-occur in a single narrative. We extend our prior work (Li et al. 2012) on acquiring script-like knowledge focusing on achieving properties of an automatically acquired domain model necessary for story generation.

The Scheherazade System

The SCHEHERAZADE system attempts to create a novel, fictional narrative about a simple, user-provided topic. For example, the user may request a story about a "bank robbery". The system uses crowdsourcing to rapidly acquire a number of linear narrative examples about typical ways in which the topic might occur. In other words, we collect human experiences in narrative form and learn a generalized model—a plot graph—about the topic domain.

Crowdsourcing is the process of breaking a complex task into multiple small tasks that can be completed quickly by people without specific training. Answers to the small tasks are then aggregated programmatically to form the solution to the original problem. Crowdsourcing utilizes collective human intelligence to solve problems that requires special expertise otherwise.

By learning plot graphs from the collected narratives, our system can generate a story about any topic for which a crowd of people can generally agree on the main events that should occur, although not necessarily on the specific sequence of events. The following sections describe our technique for automatically learning a plot graph from a human crowd and using the plot graph for story generation.

Plot Graph Learning

To learn a plot graph for a given topic we use a three-stage process. First, a corpus of narrative examples is acquired. Second, the events their precedence relations are learned. Third, mutual exclusions between events are learned. The first two steps are described in detail in (Li et al. 2012) and summarized below. One of the contributions of this paper is the identification of mutual exclusion relations between events and optional events in the domain model, critical for traversing plot graphs that contains multiple variations of

the same situation and generating coherent stories.

Acquiring the Corpus

The process begins with a user request for a story on a particular topic (e.g. going on a date, robbing a bank, etc.). If the topic is unknown to the system, a query is sent to Amazon Mechanical Turk (AMT) to solicit *typical* narratives in natural language on that topic. To simplify the complexity of natural language processing, crowd workers are requested to segment their narratives such that each sentence contains one event. Crowd workers are further instructed to use one verb per sentence, and avoid complex linguistic structures such as conditionals, compound sentences, and pronouns.

Learning Events and Precedence Relations

The system first analyzes narrative examples to discover the consensus of primitive plot points, or events. Specifically, sentences from different narrative examples that are semantically similar are clustered together to create a plot event. Thanks to the simplified language used in the narrative examples, the system can use simple semantic similarity and clustering algorithms to discover plot events with relatively high accuracy. For example, we identify plot events for the bank robbery scenario with 80.4% purity, a standard measure of cluster homogeneity. We extend the technique by using a second round of crowdsourcing in which the system asks crowd workers to correct clustering errors by (a) eliminating sentences from clusters that do not appear to semantically belong and then (b) inserting those sentences into more appropriate clusters. Preliminary results indicate that the crowd can improve the purity to 89.8%.

Next, we identify the precedence constraints between plot events. Crowd workers produce noisy and sometimes erroneous answers such as omitting steps, requiring resilience against noise. Given two events e_i and e_j, we test two hypotheses respectively representing the two orderings $e_i \rightarrow e_j$ and $e_j \rightarrow e_i$ based on a binomial distribution. Hypotheses above a confidence level T_p are accepted. We set $T_p > 0.5$ so at most one of the two hypotheses is accepted. It is possible to reject both, in which case no ordering relation is extended between the two events.

Orderings that fall slightly below the threshold may be added to the graph if adding the relation reduces an error metric, computed as the disparity of (a) the distance between two events on the graph and (b) their average distance in the narrative examples. This can be understood as local relaxation of the manually set threshold.

Mutual Exclusions and Optional Events

Extending our previous work, we identify mutual exclusion links and optional events in the plot graph. It is commonly the case that is more than one way to perform an activity. Prior techniques do not differentiate between variations and incorporate events that should never co-occur because they represent alternatives. In our plot graph representation, mutual exclusion relations indicate that two events cannot co-occur in a single narrative.

Mutual exclusion relations are identified based on *mutual information* between events, the measure of their interdependence. Suppose $E_i \in \{0,1\}$ is a random variable indicating if event e_i exists in an input narrative. The mutual information between two events is:

$$\mathrm{MI}(E_i, E_j) = \sum_{E_i \in \{0,1\}} \sum_{E_j \in \{0,1\}} p(E_i, E_j) \log\left(\frac{p(E_i, E_j)}{p(E_i)p(E_j)}\right)$$

where $p(\cdot)$ denotes the probability of the random variables. For example, $p(E_i = 1)$ is the probability that event e_i happens in a narrative, estimated as the ratio of input narratives containing e_i to the total number of narratives in the corpus. $p(E_i = 1, E_j = 1)$ is the probability that e_i and e_j happen in the same narrative, etc. We can also write:

$$\mathrm{MI}(E_i, E_j) = C(0,0) + C(0,1) + C(1,0) + C(1,1)$$

where

$$C(a,b) = p(E_i = a, E_j = b) \log\left(\frac{p(E_i = a, E_j = b)}{p(E_i = a)p(E_i = b)}\right)$$

The partial sum $C(0,1) + C(1,0)$ indicates the tendency for the two random variables to take on different values, or that the presence of one event predicts the absence of the other event. We recognize two events to be mutually exclusive when $C(1,0) + C(0,1) > 0$, indicating mutual exclusion, and $\mathrm{MI}(E_i, E_j)$ is greater than a threshold T_m, indicating strong interdependence between the two variables. It is difficult to pre-determine a good threshold. We discuss robustness against parameter selection when we discuss story generation in the next section.

For the purpose of creating narratives, we assume that precedence constraints are *necessary* in nature. That is, if e_i precedes e_j, e_j can only execute after e_i is executed because it establishes a necessary condition. If e_i and e_j are mutually exclusive and unordered relative to each other, a story generation algorithm can pick one or the other to be in the story, but not both. However, if there are ordering relations between mutually exclusive events, then we have a contradiction: e_j can neither execute before e_i (due to precedence) nor after e_i (due to mutual exclusion). This suggests our necessity assumption does not apply in this case and we should interpret e_i as being optional for the story. In other words, if e_i occurs in a narrative, e_j will be excluded, but if e_i does not occur the e_j is still necessary for its successor events. Thus, e_j is conditioned on e_i.

In order to recognize e_i as an optional event, we have an additional requirement that e_i is not also mutually exclusive with another event e_k that also precedes e_j.

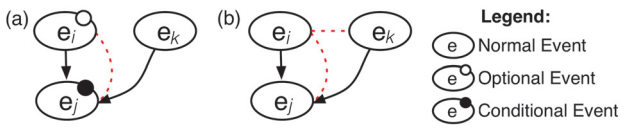

Figure 1. (a) Event A is optional and C is conditional. (b) Event A is not optional due to the mutual exclusion between A and B.

Figure 1 shows two cases for whether such an e_k exists. The existence of predecessor e_k, being mutually exclusive to e_i, indicates we have a choice between e_i and e_k, and e_j can safely happen when e_k is chosen. In this case, our assumption leads to no contradictions and we do not recognize e_i to be optional, as shown in Figure 1(b). The recognition of optional events can be considered as local relaxation of the necessity assumption.

Figure 2 shows the plot graph learned for a bank robbery. The plot graph was generated from 60 crowdsourced examples using the procedure described above. Close inspection of the plot graph reveals characters that change their behavior conditionally: the cashier (Sally) will not greet the bank robber (John) if he is wearing a mask. Mutual exclusions reveal a number of variations: pulling a gun versus handing a note; using a bag versus handling the money directly; escaping versus being caught; etc. Note that we manually corrected the semantic clusters to simulate the iterative process of crowdsourcing corrections, which was not yet fully implemented, but can achieve up to 90% accuracy with crowdsourced clustering correction, as noted earlier.

Story Generation with Learned Plot Graphs

In this section, we describe how our system generates narratives from a plot graph. A plot graph defines a space of linear event sequences that will be recognized as valid stories on the given topic. Thus, once a plot graph has been constructed the only question that remains is which of those valid stories to select and return. The system generates stories by stochastically adding events to the story such that no precedence constraints or mutual exclusion relations are violated, which is akin to a biased random walk. The story ends when one ending event is reached. Figure 3 shows the story generation algorithm.

The story generation algorithm samples the space of possible narratives by iteratively adding "executable" events to a story. An event is executable when all of its direct, non-optional predecessors have been executed, except those parents excluded by mutual exclusion

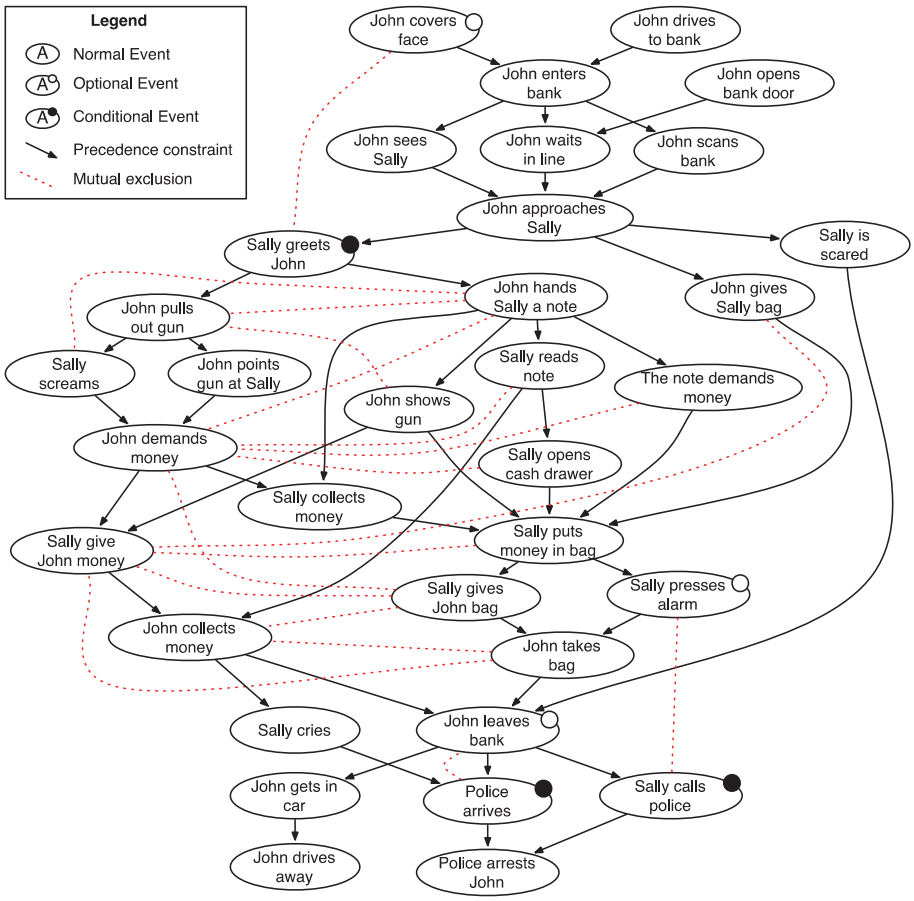

Figure 2. A plot graph for the bank robbery activity.

```
Function GENERATESTORY (plot-graph)
   For each optional event e ∈ plot-graph do
      Insert a link between each direct predecessor of e and each
            direct successor of e
   Let story ← ∅, best ← −∞
   Repeat n times
      new-story ← WALKGRAPH(plot-graph)
      value ← EVALUATEFITNESS(new-story)
      If value > best then do
         story ← new-story, best ← value
   Return story

Function WALKGRAPH (plot-graph)
   Let story ← ∅
   While not ISCOMPLETESTORY(story, plot-graph) do
      Let options ← EXECUTABLEEVENTS(plot-graph, story)
      Let e ← pick an event from options
      story ← story + e
      plot-graph ← UPDATEGRAPH(plot-graph, story)
   Return story

Function UPDATEGRAPH(graph, history)
   excluded ← events recursively excluded by mutual exclusions
   expired ← events ruled out by temporal orderings
   For each e ∈ excluded do
      Insert a link between each direct predecessor of e and each
            direct successor of e
   Return REMOVEEVENTS(excluded ∪ expired, graph)
```

Figure 3. The story generation algorithm.

relations. The event to add to the story is selected based on a distribution over executable events. The distribution could be uniform across executable events or based on the overall quality of the story once an executable event is added.

Once a plot event from the list of executable events is added to the narrative, the system performs graph maintenance to keep track of mutual exclusion relations and optional events. Events that are mutually exclusive to the newly executed event are removed from the graph. To simplify bookkeeping, we also remove optional events that have not been executed but their successors have. These optional events have been skipped and can no longer be executed without violating precedence relations. To avoid losing structural information, direct parents of removed events are linked to direct successors of removed events with precedence constraints.

As noted in the previous section, mutual exclusion relations may be unintentionally omitted due to the application of a threshold T_m. To compensate, we allow mutual exclusion relations to propagate during story generation. Specifically, during graph maintenance, we recursively remove any event that is completely dependent on deleted events, i.e. an event is removed if all its parents have also been removed. The exclusion process continues until no such events exist.

The story generation algorithm makes use of a fitness function that indicates any preferences over the space of stories represented by the plot-graph. The plot graph describes a social situation that is relatively well constrained, so the only question that remains is how prototypical should the resultant story be. We define *typicality* as a function of the likelihood of events (nodes) and of specific sub-sequences (node-link-node sequences). By varying the inclusion of nodes and links according to their likelihood while respecting the before relations, we can generate stories that are legal but with arbitrary typicality within the norm.

We have a wealth of probabilistic information to draw from as a consequence of how we learn the plot graph:

- **Typicality of events**—the probability of an event being part of a situation, P(e).
- **Adjacency**—the probability that two events should occur immediately adjacent to each other, P(e_1*e_2 | e_1 ∧ e_2).
- **Co-occurrence**—the probability that any two events have been observed in the same crowdsourced story, P(e1 ∧ e2).

The most prototypical story that can be generated from a given plot graph, for example, may be defined as inclusion of the n most probable events, ordered according to the most probable before relations between those n nodes. We can generate more interesting stories about the same topic by finding a legal sequence with (a) an unlikely event, such as "Sally cries" (occurring in 6.7% of crowdsourced bank robbery examples); (b) likely events that occur in an unlikely ordering; (c) non-adjacent events that are typically adjacent; (d) pairs of events that have low co-occurrence; or (e) omission of an event that frequently co-occurs with a present event.

The generation algorithm produces over a million legal linearizations for the bank robbery graph. The *authorial leverage* (Chen, Nelson, & Mateas 2009)—the ratio of possible narratives to authoring effort—of our system is high considering the input of 60 examples. The quality of generated stories is assessed in the next section.

Evaluation

We conducted a large-scale evaluation of the quality and coherence of stories generated from the bank robbery plot graph. We recruited human judges to read stories as a list of sentences and to make changes to make them more coherent. Judges could delete sentences by dragging them to a trash bin or reorder sentences by dragging them to the correct location. Judges could also write up to three new events, but could not specify where those events should be positioned. To detect dishonest behaviors, we insert one obviously incorrect event that must be deleted for the judge's response to be accepted.

We recruited 360 people on AMT and paid 20-30¢ as compensation. 60 participants edited the 60 stories from

Table 1. Experiment results.

	Human	Comp.
Mean original length	12.78	23.14*
Mean final length	11.82	21.89*
Mean events added	0.33	0.49§
Mean events added (normalized)	0.03	0.02§
Mean events deleted	0.30	0.76*
Mean events deleted (normalized)	0.02	0.03§
Mean events deleted (2 events withheld)	0.28	0.27§
Mean events deleted (2 withheld, norm.)	0.02	0.01§
Mean events moved	0.53	3.57*
Mean events moved (normalized)	0.04	0.15*

* The difference is statistically significant ($p < 0.05$)
§ The difference is not statistically significant ($p \geq 0.05$)

the crowdsourced corpus. 100 additional stories were uniformly sampled from all possible stories that could be generated by the plot graph; the most frequent sentence of the underlying natural language cluster was selected to describe an event. 300 participants edited generated stories such that three judges saw each story.

We take the number of edits as an indication of story quality; less edits indicate better quality. The number of events added, deleted, or moved are shown in Table 1. Welsh t-tests were used to determine if the difference between human-authored and computer generated stories is statistically significant at $p < 0.05$. Plot graphs merge elements from many stories, so computer-generated stories are on average nearly twice as long as human-authored stories. We also report mean additions, deletions, and reorderings after normalizing for story length.

We find the difference between human-authored and computer-generated stories to be insignificant in many measures, which is quite suggestive given the scale of the evaluation. No significant differences exist in number of added events. We conclude that SCHEHERAZADE does not omit essential events any more often than human authors.

We find a small but statistically significant difference in the number of events deleted from computer-generated stories. Although statistically significant, the mean difference between conditions is less than one deleted event. The significance vanishes when two events "wait in line" and "open door" are withheld. The two events account for 64.5% of deletes, and occurred rarely in the corpus (6 and 9 times out of 60), which explains why the system has less certainty about their inclusion. We conclude that despite the existence of multiple incompatible alternatives throughout the plot graph, the system does not add events that contradict human intuition. We attribute this result to the use of mutual exclusion relations to successfully separate incompatible events. The system tells "verbose" stories, containing events that are correct but unnecessary for human comprehension, which needs to be addressed by future work.

The reordering of events is a consequence of under- or over-constrained temporal precedence relations. The fact that these events are not deleted indicates that they contribute to the story but are not ideally positioned. Moves may be due to preferences for event ordering or errors in the plot graph. 32.3% of moves are consistent with the plot graph, indicating that the reordering exists in another legal story and the judge may be indicating a preference for another story that can be generated.

The rest of the moves violate temporal constraints in the plot graph, which may indicate a preference for an ordering that was over-constrained rather than a severe error. For example, pressing the alarm is over-constrained to occur in the second half of the story. Two moved events account for a plurality of inconsistencies: "get in car" is uniformly moved from end to beginning; "Sally cries" is a rare event in the corpus. Removing these two events reduces the inconsistencies to 44% of generated stories. Over-commitment to temporal relations does not necessarily imply that the plot graph or subsequently generated stories are nonsensical or incoherent.

Conclusions

In this paper, we present an *open story generation* system, SCHEHERAZADE, which tackles the challenge of creating stories about any topic that humans can agree upon how to describe. The system achieves this by first learning a model—plot graph—of the given topic from crowdsourced examples. We leverage emerging techniques for learning from stories. We extend these techniques by recognizing when events are mutually exclusive, thus ensuring the system cannot generate stories that conflate distinct variations. SCHEHERAZADE generates stories by stochastically sampling from the space of stories defined by the plot graph, finding a sequence of events that does not violate any constraints. A large-scale evaluation confirms that generated stories are of comparable quality to narratives written by untrained humans.

To date the reliance of story generation systems on *a priori* known domain models has limited the degree to which these systems can be said to possess narrative intelligence. Open story generation overcomes knowledge engineering bottlenecks by demonstrating that a system can learn to tell stories by using a crowd of anonymous workers as a surrogate for real-world experiences. We believe that this is a crucial step toward achieving human-level computational narrative intelligence.

Acknowledgements

We gratefully acknowledge the support of the U.S. Defense Advanced Research Projects Agency (DARPA). Special thanks to Alexander Zook.

References

Brenner, M. 2011. Creating Dynamic Story Plots with Continual Multiagent Planning. *Proc. of the 24th National Conference on Artificial Intelligence*.

Chambers, N. and Jurafsky, D. 2008. Unsupervised Learning of Narrative Event Chains. *Proc. of the 46th Annual Meeting of the Association for Computational Linguistics*.

Chen, S., Nelson, M., and Mateas, M. 2009. Evaluating the Authorial Leverage of Drama Management. *Proc. of the 5th AAAI Conference on AI and Interactive Digital Entertainment*.

Gervás, P., Díaz-Agudo, B., Peinado, F., and Hervás, R. 2005. Story Plot Generation based on CBR. *Journal of Knowledge-Based Systems*, 18, 235-242.

Lebowitz, M. 1987. Planning stories. *Proc. of the 9th Annual Conference of the Cognitive Science Society*.

Li, B., Lee-Urban, S., Appling, D.S., and Riedl, M.O. 2012. Crowdsourcing Narrative Intelligence. *Advances in Cognitive Systems*, 2, 25-42.

Liu, H. and Singh, P. 2002. MAKEBELIEVE: Using Commonsense to Generate Stories. *Proc. of the 18th National Conference on Artificial Intelligence*.

Meehan, J.R. 1976. *The Metanovel: Writing Stories by Computers*. Ph.D. Thesis, Yale University.

McIntyre, N. and Lapata, M. 2010. Plot induction and evolutionary search for story generation. *Proc. of the 48th Annual Meeting of the Association for Computational Linguistics*.

Nelson, M. and Mateas, M. 2005. Search-Based Drama Management in the Interactive Fiction *Anchorhead. Proc. of the 1st AAAI Conference on AI and Interactive Digital Entertainment*.

Porteous, J., Cavazza, M., and Charles, F. 2010. Applying Planning to Interactive Storytelling: Narrative Control Using State Constraints. *ACM Transactions on Intelligent Systems and Technology*, 1(2), 1-21.

Regneri, M., Koller, A., and Pinkal, M. 2010. Learning Script Knowledge with Web Experiments. *Proc. of the 48th Annual Meeting of the Association for Computational Linguistics*.

Riedl, M.O. and Young, R.M. 2010. Narrative Planning: Balancing Plot and Character. *Journal of Artificial Intelligence Research*, 39, 217-268.

Swanson, R. and Gordon, A. 2012. Say Anything: Using Textual Case-Based Reasoning to Enable Open-Domain Interactive Storytelling. *ACM Transactions on Interactive Intelligent Systems*, 2(3), 16:1-16:35.

Turner, S. 1994. *The Creative Process: A Computer Model of Storytelling*. Lawrence Erlbaum Associates.

Weyhrauch, P. 1997. *Guiding Interactive Fiction*. Ph.D. Thesis, Carnegie Mellon University.

Winston, P.H. 2011. The Strong Story Hypothesis and the Directed Perception Hypothesis. In *Technical Report FS-11-01, Papers from the AAAI Fall Symposium*. AAAI Press.