Proceedings of the Seventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment

The SAM Algorithm for Analogy-Based Story Generation

Santiago Ontañón
Artificial Intelligence Research Institute (IIIA)
Spanish Council for Scientific Research (CSIC)
Campus UAB, 08193 Bellaterra, Spain
santi@iiia.csic.es

Jichen Zhu[*]
School of Visual Arts and Design
University of Central Florida
Orlando, FL, USA 32826-3241
jichen.zhu@ucf.edu

Abstract

Analogy-based Story Generation (ASG) is a relatively under-explored approach for story generation and computational narrative. In this paper, we present the SAM (Story Analogies through Mapping) algorithm as our attempt to expand the scope and complexity of stories generated by ASG. Comparing with existing work and our prior work, there are two main contributions of SAM: it employs 1) analogical reasoning both at the specific story content and general domain knowledge levels, and 2) temporal reasoning about the story (phase) structure in order to generate more complex stories. We illustrate SAM through a few example stories.

Introduction

Story generation is an important area for interactive digital entertainment and cultural production. Built on the age-old tradition of storytelling, algorithmically generated stories can be used in a wide variety of domains such as computer games, training and education. In addition, research in story generation may shed light into the broader phenomena of human and computational creativity (Gervás 2009). Compared with the established narrative forms such as prose fiction, however, computer-generated stories are still in its early stage. Despite the recent progress in the area, such as in planning-based approaches (Meehan 1976) and multi-agent simulation-based approaches (Theune et al. 2003), these stories are still fairly rudimental in terms of both the depth of meanings and the range of their varieties.

Different story generation techniques have specific built-in narrative affordances and constraints. For instance, the planning-based story generation approach lends itself very well to stories that are action-based, goal-driven with strong *causal connections* between events. By comparison, computational analogy-based story generation (ASG) is a relatively under-explored research direction that can generate stories where events are inter-connected based on their intrinsic *associations*. In this paper, we continue our existing work on ASG, focusing on expanding the scope and complexity of the stories our system can generate. More specifically, we present the SAM (Story Analogies through Mapping) algorithm, which generates stories by transferring story content from a complete source story to a partial target story.

The approach presented in this paper builds on the work done in the Riu system (Ontañón and Zhu 2010), and generalizes the latter in two key aspects: first, SAM allows the generation of longer stories with more complex time structures; second, SAM allows the inclusion of domain knowledge in the process of story generation. Comparing with other existing systems, there are two main contributions of SAM: it employs 1) analogical reasoning both on the specific story content level and the general domain knowledge level, and 2) temporal reasoning about the story (phase) structure in order to generate more complex stories.

This paper is organized as follows. We first provide background on story generation and computational analogy. We then formally describe the SAM algorithm and show several example stories generated by SAM. Then, we compare SAM with other story generation systems andwe conclude the paper with discussion and future directions.

[*]Jichen Zhu is currently at Drexel University, Philadelphia, PA.

Background

Automatic story generation is an interdisciplinary topic focusing on devising models for algorithmically structuring and producing narrative content and/or discourse. In general, story generation systems can be classified into three main categories (Bailey 1999): character-centric, author-centric and story-centric, depending on whether the systems focus on simulating characters, simulating the authorial process, or reasoning about the story structure itself.

- Character-centric systems generate stories by simulating characters in a world. Examples in this category include Tale-spin (Meehan 1976) and the Virtual Storyteller (Theune et al. 2003).
- Author-centric systems, such as the MEXICA system (Pérez y Pérez and Sharples 2001), model the author's throught process during the process of story-writing.
- Story-centric systems, such as the Fabulist (Riedl 2004), generate stories by modeling the structural properties of the stories themselves.

Different techniques have been studied in story generation, the most common of which is automated planning.

Salient examples of planning-based story generation systems include Tale-spin (Meehan 1976), Universe (Lebowitz 1984) and Fabulist (Riedl 2004). By contrast, computational analogy algorithms have not been sufficiently explored in the domain of story generation.

As a model of the human cognitive process of analogy-making, computational analogy operates by identifying similarities and transferring knowledge between a source domain S and a target domain T. The intuitive assumption behind analogy is that if two domains are similar in certain key aspects, they are likely to be similar in other aspects. Given a target domain T, this process is composed of four stages (Hall 1989): 1) *recognition* of a candidate analogous source, S; 2) *elaboration* of an analogical mapping and inferences between source domain S and target domain T; 3) *evaluation* of the mapping and inferences, and; 4) *consolidation* of the outcome of the analogy for other contexts (i.e., learning). In this paper, we demonstrate how SAM exploits the elaboration stage of computational analogy.

Existing computational analogy approaches can be classified into three classes: symbolic, connectionist and hybrid models; an in-depth overview can be found at (French 2002). SAM internally uses the Structure Mapping Engine (SME) algorithm (Falkenhainer, Forbus, and Gentner 1989), a symbolic approach, although SAM can be generalized to use any other computational analogy algorithm.

SAM grew out of the Riu system (Ontañón and Zhu 2010), a text-based interactive narrative system that uses story generation to narrate stories about the connection between two parallel worlds — a real world and a memory world. One of the challenges we had in Riu was the limited scope and complexity of the stories that the ASG component of the system can generate. As a result, SAM reflects our latest effort to extend analogy-based story generation towards a more mature narrative form.

SAM: Story Analogies through Mapping

This section presents the SAM algorithm for story generation. Given a complete source story S, and an incomplete, target, story T, the goal of SAM is to generate a new story R by transferring knowledge from S in order to complete T. Let us first introduce the story representation formalism.

Story Representation

The representation formalism used by SAM is an extension of that in our prior work in ASG (Ontañón and Zhu 2010). After briefly summarizing our existing representation, we will focus on the essential modifications and generalizations developed in order to expand the scope and complexity of the stories generated by our ASG system.

As in our prior work, a basic story S is represented as a set of *phases*, P_S, where each phase represents a different instant of time. The story elements in a phase are represented in two parallel forms: a *computer understandable* description (CUD) and a *human understandable* description (HUD). The CUD is a frame-based representation of the phase, whereas the HUD consists of a collection of pre-authored natural language phrases and sentences. The HUD captures explicit knowledge about the story author's preferred instantiations of a given story in natural language. The CUD and HUD are linked, so that when SAM employs analogical reasoning to manipulate the CUD, it can also manipulate the HUD accordingly (to some extent) to produce the resulting story in natural language.

Fig. 1 illustrates the above elements in a basic story. The story is about a robot character, Ales, who had a pet bird when he was young. It has two phases; in the first phase, Ales plays with the bird, and in the second, the bird dies and Ales is sad. The nodes in the CUD representing entities (Ales, the bird, etc.) are shown as dark grey ovals and relations between entities are shown as white boxes. An important part of the CUD relations are drawn from *force dynamics* (Talmy 1988) (e.g. agonist, stronger, antagonist, move-tendency) to annotate the force relationship between different entities. Elsewhere (Ontañón and Zhu 2010), we have demonstrated that the force dynamics annotations are useful to identify deep analogies between stories and improve the results of ASG.

Two main extensions to this story representation were done in order to generate more complex stories. First, we generalize the definition of the *phase structure*. Instead of assuming that all phases are linearly sequenced, we generalized to a directed acyclic graph (DAG) to represent a wider range of how phases can be organized, such as branches and alternatives (e.g. stories with multiple alternative endings). Nodes in the DAG represent phases, and there is an edge from a phase p_1 to another phase p_2, when p_2 can happen immediately after p_1. When there is a path from a phase p_1 to another phase p_2, we say that p_1 *precedes* p_2. When there is no path from p_1 to p_2 nor from p_2 to p_1, we say that p_1 and p_2 *exclude* each other, since one cannot happen if the other does. In the example shown in Fig. 1, the phase structure has two nodes, and Phase 1 precedes Phase 2.

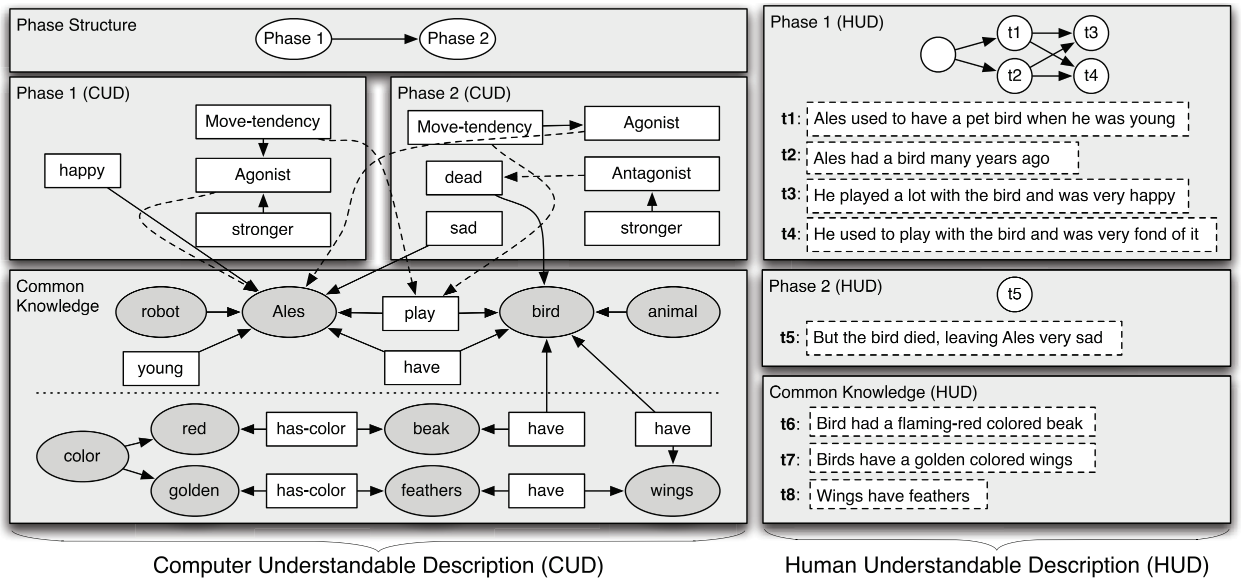
The second main extension is the addition of background knowledge. Our prior representation included a special phase, called "common", where entities and relations that are shared among all the phases can be specified to avoid replication. We extended this idea to include additional domain knowledge about the entities and relations potentially relevant to the stories, and called it *common knowledge*, which is also divided in a CUD and a HUD.

In the example shown in Fig. 1, background knowledge is shown below the dotted line in the common knowledge box. In this case, we have specified that birds have a beak and wings, and that wings have feathers. The more additional knowledge specified, the better SAM will be able to find analogies between stories, as we illustrate later.

The SAM Algorithm

In this section we will present the SAM algorithm[1]. SAM takes three input parameters: T, S, and m_i. T and S are the target and source stories, and m_i is a phase mapping from T to S (which is optional). $SAM(T, S, m_i)$ returns a new story R, resulting from extending T by analogy with

[1]SAM can be downloaded from: https://sites.google.com/site/santiagoontanonvillar/software



Phase Structure
Phase 1
Phase 2
Phase 1 (CUD)
Move-tendency
happy
Agonist
stronger
Phase 2 (CUD)
Move-tendency
Agonist
dead
Antagonist
sad
stronger
Common Knowledge
robot
Ales
play
bird
animal
young
have
red
has-color
beak
have
have
color
golden
has-color
feathers
have
wings
Phase 1 (HUD)
t1
t2
t3
t4
t1: Ales used to have a pet bird when he was young
t2: Ales had a bird many years ago
t3: He played a lot with the bird and was very happy
t4: He used to play with the bird and was very fond of it
Phase 2 (HUD)
t5
t5: But the bird died, leaving Ales very sad
Common Knowledge (HUD)
t6: Bird had a flaming-red colored beak
t7: Birds have a golden colored wings
t8: Wings have feathers
Computer Understandable Description (CUD)
Human Understandable Description (HUD)

Figure 1: Complete representation of a basic story.

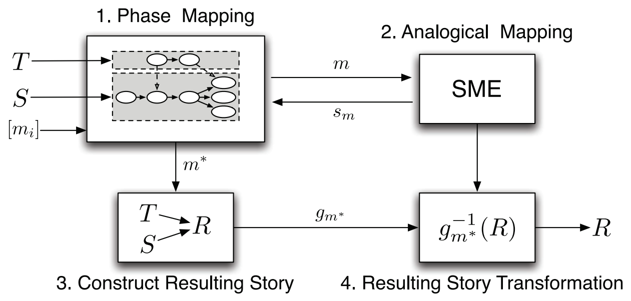


Figure 2: The four steps of the SAM algorithm.

S. Internally, SAM uses Structure Mapping Engine (SME) (Falkenhainer, Forbus, and Gentner 1989) to generate analogical mappings between T and S. The execution of SAM consists of four main steps, illustrated in Fig. 2:

1. **Generate all possible phase mappings**: Let P_T and P_S be the sets of phases of the two stories, and given an injective mapping m from P_T to P_S (i.e. a mapping such that each element of P_T is mapped to one of P_S, and in which not two elements of P_T are mapped to the same element in P_S), we say that the mapping *is consistent* if for each pair of phases $p_1, p_2 \in P_T$:

   - If p_1 precedes p_2, then $m(p_1)$ precedes $m(p_2)$
   - If p_1 and p_2 exclude each other, then $m(p_1)$ and $m(p_2)$ exclude each other.

   If a phase mapping m_i is specified as input parameter, then $M = \{m_i\}$, otherwise, SAM computes M as the set of all the possible consistent injective mappings from P_T to P_S. An example consistent injective mapping between two stories is shown in Fig. 3.

2. **Find the analogical mappings**: for each phase mapping $m \in M$, SAM does the following:

   - Let $P_S^m = \{p \in P_S | \exists p' \in P_T : m(p') = p\}$, i.e. all the phases from S in the mapping m.
   - e_S^m is constructed as all the entities in the CUDs of the phases in P_S^m and in the common knowledge of S. e_T is defined as all the entities in the CUDs of T.
   - r_S^m is constructed as all the relations in the CUDs of the phases in P_S^m and in the common knowledge of S. r_T is defined as all the relations in the CUDs of T.
   - SME is called using $e_T \cup r_T$ as the target domain and $e_S^m \cup r_S^m$ as the source domain. SME returns two things: an analogical mapping g_m from the target domain to the source domain, and a numerical score s_m.

   $m^* \in M$ is selected as the phase mapping in M that maximizes s_m. If M is empty, m^* will not be defined, and SAM will return an error token.

3. **Construct a resulting story** R: a new story R is constructed in the following way:

   - The phase structure DAG of R is copied from S.
   - The set of phases in R is $P_R = P_T \cup (P_S \setminus P_S^{m^*})$. The CUD of the common knowledge of T is added to all the phases in P_R that came from P_T, and the CUD of the common knowledge of S is added to all the phases in P_R that came from P_S.
   - The common knowledge in R is empty.

4. **Transform** R **using the analogical mappings**: The reverse of the analogical mapping g_{m^*} is applied to all the phases in P_R. For each phase $p \in P_R$, the following two steps are executed:

   - For each entity or relation $e \in p$ such that $\exists e' \in S_T$: $g_{m^*}(e') = e$, we substitute e by e' in p.

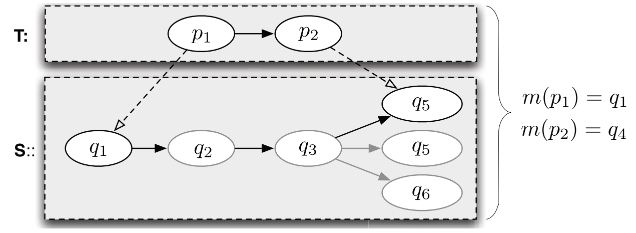


Figure 3: A phase mapping between target T and source S.

- Each time we substitute an element $e \in p$ by another element e', using the links between the CUD and the HUD, we substitute the corresponding sentence fragment of e in the HUD of p by the corresponding sentence fragment of e'.

In all the phases in R that come from P_S, if there are any sentences, entities or relations, that are not linked in some way to any of the elements or relations that belong to the mapping g_{m^*}, those are removed. This last step prevents transferring irrelevant information to the generated story. An illustration of this process is shown in Fig. 4, where we can see parts of the CUD and HUD of a phase, a mapping g generated by SME, and the result of transforming the phase using such mapping is shown.

The previous ASG component in Riu was not capable of reasoning about the phase structure. Riu's ASG component assumed that T had only one phase, and the source story S had always two phases, the first of which was always mapped to the only phase in T. In comparison, SAM affords much more flexible mapping between temporal phases in S and T and thus more complex mappings between events. For instance, if the last phases of S are mapped to the first phases of T, SAM can project *backwards* in time by inferring what kind of past events lead to the current situation in T (which Riu could not do). Another example is that SAM can fill-in the "temporal holes" between phases in T.

Additionally, SAM uses general domain knowledge, which could not be used in Riu. In its Step 2 and 4, SAM uses domain knowledge, in addition to specific story content knowledge. This allows the system to construct analogical mapping and transform knowledge from S to T that may not be explicit in the story itself.

Instead of providing a source story to SAM, SAM could also be used in conjunction to a retrieval mechanism using a repository of source stories, and letting the retrieval mechanism decide which source story to use.

Examples

In this section we will show two stories generated by SAM using stories developed for the Riu system as source and target stories. The first example will serve to illustrate the analogical mapping and resulting story transformation steps of SAM (Steps 2 and 4), and the second one to illustrates the two main contributions of SAM with respect to our prior work in Riu: the phase mapping process (step 1) and the utility of background knowledge.

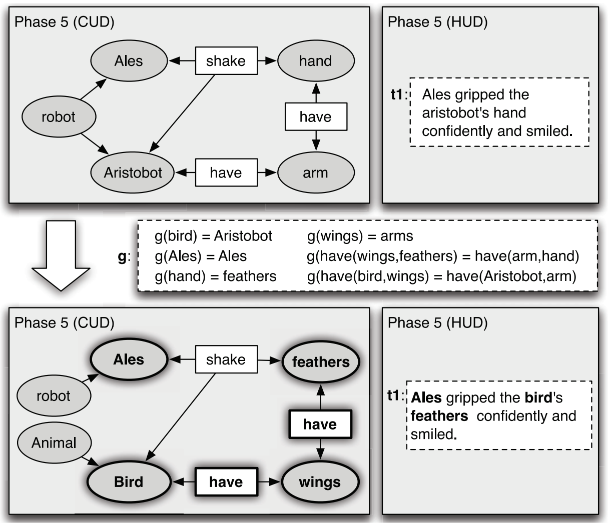


Figure 4: Transformation of a phase in the resulting story by reversing the analogical mapping g found by SME.

Using the following source story:

> *Ales remembered the garage in which he had his first oil change, it was all red. His owners said he was rusty, and forced him to change his oil, he was a fool to accept. Ales felt very awkward afterwards, and decided that he would have to be really rusty before the next time he gets an oil change. He wondered why no one ever complained about oil changes.*

and the following target story:

> *One day, Ales was walking in an alley, when he saw a cat in front of him. Ales hesitated about what to do with the cat since he was late for work. Ales played with the cat.*

SAM generates the following story by inserting an extra phase after the target story:

> *One day, Ales was walking outside, when he saw a cat in front of him. Ales hesitated about what to do with the cat since he was late for work. Ales played with the cat. Ales felt very awkward afterwards, and decided that Ales would have to be really rusty before the next time Ales played with the cat. Ales wondered why no one ever complained about that.*

To generate this story, Step 2 of SAM maps elements from the source to the target using SME. For instance, "getting an oil change" was mapped to "playing with the cat," as we can see in the generated story. Therefore, during Step 4, SAM took the second phase of the source story, and replaced the appearances of "getting an oil change" by "playing with the cat." Analyzing the CUD of the generated story, we can see SAM found other interesting mappings that were not reflected in the output text such as "owners" to "work" (because both are pushing Ales to do something against its will, which can be detected thanks to force dynamics).

The above example involves a source story with 2 phases, and a target story with a single phase. SAM can generate much more complex stories, by using more complex source and target stories, thanks to being able to reason about the phase structure and to allow the inclusion of domain knowledge. For example, by using the story in Fig. 1 as the target, and the following story as the source:

Around Ales the street was grimy with morning rain and industrial soot. He'd never seen anything like the city before, it was his first day in Polis. There had to be a job somewhere around here for him!
'Look out, a thief!', Ales heard himself calling. Ahead, the aristobot, whirled and spotted the would-be pick-pocketer he'd been warned of. He tossed a quick glower in the retreating thief's direction The aristobot then cast his hands in affectionate greeting towards Ales.
'Come here, my boy!'
With a slight sense of reluctance, Ales approached the aristobot, whose steel-brushed top hat alone represented more wealth than Ales had ever seen.
'Octavion Alcorn, in your debtedness Ales,' thrusting his hand forward in a gesture of greeting. Ales passively shook the aristobot's hand, self-conscious of the difference between his limp squeeze and Octavion's vice-like grip. Octavion politely listened as Ales explained his recent arrival to the city, all the while using an exquisite microfiber handkerchief to buff the jewel-encrusted palm he'd shaken with.
'You've done me a service, sir, now let me provide one in turn,' the aristobot said, when Ales seemed finished. He scribbled a coordinate and handed it to Ales. That's his factory, the aristobot explained, go there and Ales can find a job.

SAM generates the following story:

Ales used to have a bird when he was young. Ales used to play with the bird and was very fond of it.
'Look out, a thief!', Ales heard himself calling. Ahead, bird, whirled and spotted the would-be pick-pocketer bird'd been warned of. bird tossed a quick glower in the retreating thief's direction bird then cast his wings in affectionate greeting towards Ales.
'Come here, Ales!'
With a slight sense of reluctance, Ales approached bird, whose flaming red-colored beak alone represented more wealth than Ales had ever seen.
'bird, in your debtedness Ales,' thrusting his feathers forward in a gesture of greeting. Ales passively shook bird's feathers, self-conscious of the difference between his limp squeeze and bird's vice-like grip. bird politely listened as Ales explained his recent arrival to the city, all the while using an exquisite microfiber handkerchief to buff the jewel-encrusted feathers he'd shaken with.
'Ales've done bird a service, sir, now let bird provide one in turn,' the bird said, when Ales seemed finished. bird scribbled a coordinate and handed it to Ales. That's his cage, bird explained, go there and Ales can find a job.

In this example, the source story has 6 phases, and the target has 2, the phase mapping m^* used to generate the story is exactly the one shown in Fig. 3. Notice that the DAGs representing the phase structure of each story allows SAM to automatically align the phases of the two stories in a way that respects the temporal order of events of both input stories. Moreover, notice that the second phase of the target story (when the bird dies) is not present in the generated story, this is because such phase was found to be analogous to phase q_4 of the source (as shown in Fig. 3). When generating the resulting text, phases q_4, q_5 and q_6 are alternatives and one is chosen at random. If q_4 was selected, the bird would've died before handing the coordinates to Ales, leaving Ales very sad.

This second example illustrates the usefulness of having domain knowledge. Giving SAM knowledge that birds have wings and that there are feathers in the wings allows SAM to do the analogy that the wings are like the arms of the *aristobot*, and the feathers are like its hands. Fig. 4 shows this mapping, and how a portion of a phase was transformed using it. If we were to remove that part of the background knowledge, then, Ales would shake the bird's hands, instead of the bird's feathers in the resulting story. The more information in the background knowledge in the input stories given to SAM, the better the mappings that can be found, and the better the stories generated by SAM. Domain knowledge is also responsible for realizing that factory should be replaced by cage, as it allows SME to map factory to cage.

Comparison with Other Approaches

The most related algorithm to SAM is the *Story Translator* (Riedl and León 2009). It uses analogy as the main generative method and planning to fill in the gaps in the analogy-generated content. The input to this system is a story represented as a plan and two domain models. The latter contain the set of objects and planning operators available in each domain. The CAB algorithm (Larkey and Love 2003) is used to find a mapping between the two domain models, and this mapping is then used to translate the input story from one domain to the other, filling the gaps using planning in case the mapping is not complete.

Notice that the operation of the Story Translator can be modeled in SAM (except for the final planning step to fill-in the gaps) by giving SAM two stories in the following way: the target story has no phases, but just a common knowledge component, containing all the entities, and actions available in the target domain, and giving as source a complete story, accompanied with the same kind of common knowledge component. In that way, SAM would just return the source story, but transformed using the analogical mappings found between the target and source stories (as the Story Translator does). The main difference between SAM and the Story Translator is thus that SAM generates analogies between domain models (background knowledge) and specific stories, whereas the Story Translator only computes analogies between domain models. Another difference is that SAM directly generates text; the Story Translator returns its generated stories in the form of plans.

Let us now compare with case-based reasoning (CBR) models of story generation. CBR is a common technique used in story generation systems, which shares some basic operating principles with computational analogy. MINSTREL (Turner 1993) is a generic model that generates stories by executing TRAMS (Transform Recall Adapt Methods). Some of those TRAMS, like the "Cross-Domain-Solution" TRAM, use computational analogy. In particular, given a problem (an incomplete story) in a domain D_1, the TRAM finds another domain D_2, then an analogical mapping between D_1 and D_2 (similar to the way the Story Translator does), and then maps the story from D_1 to D_2, solves the problem, and then maps back the result from D_2 to D_1. In the same way as with the Story Translator, MINSTREL's TRAMS that use analogy, do so at the domain definition level, rather than at the specific story level.

By contrast, MEXICA (Pérez y Pérez and Sharples 2001) generates stories by adding one action at a time to a given story. In order to select the next action to add, MEXICA retrieves, from a story repository, a past story that is the most similar to the current state of the story. This process of comparison can be seen as trying to find an analogical mapping between the current story state, and the past stories. Therefore, while the Story Translator and MINSTREL find analogies at the domain definition level, MEXICA finds them between specific story states (called Story-World Contexts, or SWCs), which are equivalent to the *phases* in SAM. In contrast with those systems, SAM uses both domain definitions (background knowledge) and specific story states or actions (in the phases) to find analogies between the target and the source stories.

Comparing SAM to planning-based systems (such as Tale-spin (Meehan 1976)), the latter have the advantage that the author can specify the initial and ending state of a story, and thus have a lot of control of the generated story. In ASG systems, such control is exerted by providing different source stories. The source story in ASG determines both the way the generated story will unfold, as well as the narrative style of the generated story. Stories generated by planning focus on actions and change, since each planning operator corresponds typically to an action executed by a character. ASG systems, on the other hand, do not have this bias. If the source story is very action-based, then the resulting story will be so, but if the source story is very descriptive, the same will happen with the resulting story. In terms of knowledge engineering needed, planning systems require a domain model with a complete planning operator definition, and ASG systems require a collection of source stories. Both of which require significant effort.

Conclusions and Future Work

This paper has presented the SAM story generation algorithm. SAM extends our previous work (Ontañón and Zhu 2010) in ASG in two key aspects: 1) it allows the generation of arbitrarily long stories thanks to being able to reason about the phase structure of stories; and 2) it can exploit both specific story content as well as general background knowledge in order to find analogies between stories.

We have compared SAM with existing story generation systems and discussed some of their differences and similarities. We have also seen that analogy-based story generation systems, like SAM, can generate stories which have a different aesthetic range from stories generated by other approaches, like planning.

As part of our current and future work, we are working on improving the automatic text generation capabilities of SAM. In some cases, the modifications SAM performs in the HUD due to a change in the CUD produce grammatically or semantically incorrect text. In our current work, we are working on a case-based reasoning system, capable of modifying the HUD while maintaining grammatically correct sentences, and taking into account some semantic constraints. Additionally, we are preparing an empirical user studies to evaluate the quality of stories generated by SAM.

Acknowledgements We thank J. Hunter Sizemore for authoring the stories used for evaluation. Research partially funded by the project *BUSCAMEDIA*, CEN-20091026.

References

Bailey, P. 1999. Searching for storiness: Story-generation from a reader's perspective. In *Proceedings of the AAAI Fall Symposium on Narrative Intelligence*.

Falkenhainer, B.; Forbus, K. D.; and Gentner, D. 1989. The structure-mapping engine: Algorithm and examples. *Artificial Intelligence* 41:1–63.

French, R. M. 2002. The computational modeling of analogy-making. *Trends in Cognitive Sciences* 6(5):200–205.

Gervás, P. 2009. Computational approaches to storytelling and creativity. *AI Magazine* 30(3):49–62.

Hall, R. P. 1989. Computational approaches to analogical reasoning: a comparative analysis. *Artificial Intelligence* 39(1):39–120.

Larkey, L. B., and Love, B. C. 2003. Cab: Connectionist analogy builder. *Cognitive Science* 27(5):781–794.

Lebowitz, M. 1984. Creating characters in a story-telling universe. *Poetics* 13:171–194.

Meehan, J. 1976. *The Metanovel: Writing Stories by Computer*. Ph.d., Yale University.

Ontañón, S., and Zhu, J. 2010. Story and Text Generation through Computational Analogy in the Riu System. In *AIIDE*, 51–56. The AAAI Press.

Pérez y Pérez, R., and Sharples, M. 2001. Mexica: A computer model of a cognitive account of creative writing. *Journal of Experimental and Theoretical Artificial Intelligence* 13(2):119–139.

Riedl, M., and León, C. 2009. Generating story analogues. In *AIIDE 2009*. The AAAI Press.

Riedl, M. 2004. *Narrative Generation: Balancing Plot and Character*. Ph.D. Dissertation, North Carolina State University.

Talmy, L. 1988. Force dynamics in language and cognition. *Cognitive Science* 12(1):49–100.

Theune, M.; Faas, E.; Nijholt, A.; and Heylen, D. 2003. The virtual storyteller: Story creation by. In *Proceedings of the Technologies for Interactive Digital Storytelling and Entertainment (TIDSE) Conference*, 204–215.

Turner, S. R. 1993. *Minstrel: a computer model of creativity and storytelling*. Ph.D. Dissertation, University of California at Los Angeles, Los Angeles, CA, USA.