Imaginative Recall with Story Intention Graphs

Sarah Harmon and Arnav Jhala

Department of Computer Science, University of California at Santa Cruz
Santa Cruz, USA
jhala@soe.ucsc.edu, smharmon@ucsc.edu

Abstract

Intelligent storytelling systems either formalize specific narrative structures proposed by narratologists (such as Propp and Bremond), or are founded on formal representations from artificial intelligence (such as plan structures from classical planning). This disparity in underlying knowledge representations leads to a lack of common evaluation metrics across story generation systems, particularly around the creativity aspect of generators. This paper takes Skald, a reconstruction of the Minstrel creative story generation system, and maps the representation to a formal narrative representation of Story Intention Graphs (SIG) proposed by Elson et al. This mapping facilitates the opportunity to expand the creative space of stories generated through imaginative recall in Minstrel while maintaining narrative complexity. We show that there is promise in using the SIG as an intermediate representation that is useful for evaluation of story generation systems.

1998 ACM Subject Classification I.2.7 Natural Language Processing: Discourse

Keywords and phrases Story generation, computational creativity, narrative, story intention graph

Digital Object Identifier 10.4230/OASIcs.CMN.2015.72

1 Introduction

Storytelling and creativity are key aspects of human cognition. While much work has been done on computational narrative generation, the focus of this research in recent years has been more toward generation of coherent sequences of events. Minstrel, one of the earliest story generators, utilized a case-based reasoning approach to incorporate a model of human creativity [17]. In this paper, we extend a contemporary rational reconstruction of Minstrel called Skald [16] by organizing and labeling story events. We then present a mapping between the underlying story representation in Skald to the Story Intention Graph (SIG) formalism proposed recently by [4], which is rooted in story understanding. This mapping and extensions to Skald allow us to identify areas of research that are unexplored both in terms of storytelling and creative systems.

Minstrel relies heavily on a library of cases, and employs a boredom mechanic which, although designed to generate more interesting results, quickly exhausts its library of reference stories. Considerable manual authoring is thus required as part of the original Minstrel system. There is also, notably, no reliable bridge towards a natural language generation system for a generic Minstrel-like program. As such, current attempts to expand the creative power of Minstrel produce graphs, rather than text which reads like a natural story [16]. Finally, it is difficult to compare storytelling systems like Minstrel with each other, because there is no definitive standard designed to assess the quality or scope of generated creative content. Here, we propose that a semantic representation system – the Story Intention Graph (SIG) model [4] – be used as a formalized standard of narrative meaning and comprehension.

6th Workshop on Computational Models of Narrative (CMN'15).
Editors: Mark A. Finlayson, Ben Miller, Antonio Lieto, and Remi Ronfard; pp. 72–81

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

With the adoption of this standard, generated narrative content, such as that composed by Minstrel, can be more easily analyzed, upgraded, and rewritten as natural text.

The SIG formalism provides several affordances that improve the richness of representation of stories beyond the parameterized case frames of situations. First, it is based on a rich model of internal states of agents involved in the narrative using a theory of mind approach. This approach maintains local coherence for characters while ensuring global coherence of the overall narrative. Second, it has a notion of a *plot unit* but at a richer level of semantic interconnections across plot units. Finally, the SIG representation provides a way to detect and reason analogies through metrics derived from the encodings. This is an important affordance, particularly for CBR-based generation systems.

The overall contributions of this work are two-fold. The primary contribution is the implementation of the SIG formalism in a case-based story generation system. The secondary contribution is the implementation of extensions to Minstrel's generation process in terms of event ordering and using a richer story representation to increase the expressive range of creative stories generated by the system.

2 Related Work

One of the first automated storytelling systems known was a murder mystery generator called Novel Writer [9]. The domain of generated stories for Novel Writer was very small: only one type of story was generated, and always involved a murderer, a motive, and someone who revealed the murderer. Further, the Novel Writer ruleset was highly constraining – allowing, for instance, only four possible motives for murder – and prevented the overall system from reaching a high level of creativity and expression.

Several years later, a system called TALE-SPIN [10] took a character-driven approach to story generation. In TALE-SPIN, multiple characters could develop plans to pursue individual-level goals. Additionally, characters had personalities and dynamic relationships with each other. Although revolutionary in terms of its character planning system, TALE-SPIN was criticized for not providing a model for the author's creative process and goals.

The AUTHOR program [3] was created for precisely this purpose. AUTHOR generated stories by simulating the intentions of a human author and striving to satisfy them. However, AUTHOR was designed with the underlying assumption that all generated narrative sequences must conform to a strict ruleset detailing story parameters and narrative structure. Within the AUTHOR system, then, there is not much freedom in terms of computational creativity.

The focus of modern systems is specifically on generation of plot structures (in plan-based approaches), drama management for sequencing predefined beat structures, or manipulating surface level discourse elements like language and visuals. The goal in these systems is either coherence of stories or management of player experience. While outputs of these generators do qualify as being creative, it is difficult to evaluate the systems in terms of creativity due to the variety of underlying representations and lack of an explicit model of creativity. Detailed review of modern storytelling systems is outside the scope of this paper as the primary focus is a discussion of creativity within a rational reconstruction of the classic story generation system.

3 Research Foundation

3.1 Minstrel, a Case-Based Reasoning Approach

Turner created the Minstrel [17] story generation system that takes a case-based reasoning approach to creative authoring of stories. Minstrel is a LISP program that simulates the

Table 1 A quantitative comparison between Minstrel Remixed and Skald. By using weighted TRAM searching and a modified boredom algorithm, Skald optimized TRAM results in terms of speed and retrieval quality.

Measure	Minstrel Remixed	Skald
TRAM search failure rate	19%	3.5%
Average number of TRAMs tried per search	58	16
Average number of TRAMs used when no direct match found	2.4	1.4

actions of a human author in order to produce stories. In particular, Minstrel models the human creative process by transforming memories of known events (case base) to formulate new scenarios via generalization and adaptation (referred to as *imaginative recall* in the original Minstrel description). Story elements are defined by schemas (case frames) and stored in a searchable database, and creating small changes in these schemas results in new stories.

To create new stories from prior examples, Minstrel relies on twenty-five heuristics called TRAMs ('Transform-Recall-Adapt Methods'). As an example, Minstrel contains a default TRAM called 'Standard-Problem-Solving' which simply looks for a pre-existing solution in memory. If no solution exists, the TRAM fails. The TRAM also fails if any found solutions have already been used, because such solutions are deemed 'boring' by the Minstrel system. Whenever a given TRAM fails, the problem must be transformed and Minstrel must look for a case that best matches the newly transformed problem.

3.2 Skald: Improving Minstrel's imaginative recall system

Skald[15] was developed to make the Minstrel system more robust and useful as a general-purpose story generator. While Minstrel applied TRAMs randomly, Skald employs a weighted TRAM searching algorithm which gives preferences to TRAMs that best match the original query. This technique reduces the search space, resulting in faster and higher quality generations (refer to Table 1). Skald also modifies Minstrel's boredom algorithm by only fractionally decrementing boredom signature values, enabling signatures to refresh over time and be reused in later stories. Although more 'interesting' stories are not forcibly produced as quickly as they would be in Minstrel, this technique traverses through the story library more slowly and makes more efficient use of the searchable domain. More stories can thus be produced with less manually-authored templates.

In Skald, groups of *symbols*, the most basic story elements, are grouped into *frames*. Frames may contain empty or unknown symbols (refer to Table 2). Groups of frames form an output story graph. Story characters have mental target objectives called *goals*, physical actions called *acts*, and *states*, which are results of action. Similar to Minstrel, Skald retrieves and executes author-level plans (ALPs) as part of the story generation process. Ultimately, the system constructs a connected graph with story frames as nodes, as depicted in Table 2. Most commonly, these frames are a trio consisting of a goal which *plans* an act, which, in turn, *intends* a state to occur, and wherein the state ultimately *achieves* the goal. Many of the narratives that Skald generates are formed by combining and connecting similar frame trios.

Despite being an adaptation of the original Minstrel system, Skald follows the same core ideas of simulating the human authoring process. For this reason, Skald is a suitable creative narrative generator to formalize with SIGs because it represents a valid model of computational creativity and is openly available for development. We claim that SIGs

Table 2 An example narrative generated by Skald ('Story A'). The story frames have been manually ordered and translated into natural text for readability. Each frame is composed of symbols, which may be empty, unknown, or contain a specified value.

Natural Language Equivalent	Story Frame
Frederick, the knight, did not want to be injured.	stayhealthy -> (goal) Map(*actor* -> Frederick(Knight), *object* -> Frederick(Knight), *scale* -> <empty slot>, *to* -> <empty slot>, *type* -> "Healthy", *value* -> <empty slot>)
But Fafnir, a dragon, hated Frederick.	hates -> (state) Map(*actor* -> Fafnir(Dragon), *object* -> <empty slot>, *scale* -> "Strong", *to* -> Frederick(Knight), *type* -> "Affect", *value* -> "Negative")
So, Fafnir wanted to injure him.	wantinjure -> (goal) Map(*actor* -> Fafnir(Dragon), *object* -> Frederick(Knight), *scale* -> <empty slot>, *to* -> <empty slot>, *type* -> "C-Health", *value* -> "Injured")
He fought Frederick by blowing a magical flame at him.	attack -> (act) Map(*actor* -> Fafnir(Dragon), *from* -> <empty slot>, *object* -> Flame(Magic), *to* -> Frederick(Knight), *type* -> "Fight")
Frederick was injured by the flame. His plan to stay healthy had been thwarted by Fafnir the Dragon.	injured -> (state) Map(*actor* -> Frederick(Knight), *object* -> <empty slot>, *scale* -> <empty slot>, *to* -> <empty slot>, *type* -> "Health", *value* -> "Injured")

are appropriate for three reasons, namely, they (1) provide a formal representation that can facilitate comparison between story generators beyond Skald, (2) are a bridge towards improved natural language generation in Skald and other generators, (3) expand the library of Skald without additional manual authoring.

3.3 The Story Intention Graph as a Formalism for Imaginative Recall

The SIG model provides formal, concise, and expressive [5] representations for computer-generated narratives. A shared, growing corpus of over one hundred encodings is currently available to describe and investigate narrative structures. By translating stories into SIG encodings, we have a means of expressing the diversity of structures and relationships that can be created by automated narrative generators. The discourse relations defined by SIGs

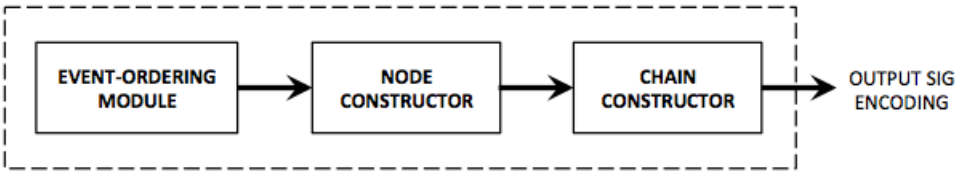

Figure 1 Block diagram of a Skald-to-SIG conversion system.

are useful in corpus annotation as well as algorithmic treatment, particularly related to analogical reasoning. A key aspect of case-based reasoning systems is the distance function used to identify similar cases during the recall phase. Current CBR-based story generators take a parameterized generalization of situations and compute a direct frame comparison to recall cases. To scale such a representation requires significant addition of semantic information to case frames, including a richer distance function to find appropriate cases from the library. Further, the transformation processes mostly generalize at the level of a single parameter's domain constraints. It has been shown [4] that the SIG formalism outperforms other representations in finding not only analogical stories individually, but also analogical sub-sets through a comparison on isomorphic sub-graphs to common SIG patterns.

The SIG model is an encoding of narrative that forms a *semantic network*. Such networks are commonly utilized in cognitive psychology for narrative comprehension studies with humans [7]. In plan-based narrative generation systems, such encodings are used within representations of plan operators and heuristic functions to search for stories [2, 1, 12]. In work related to common sense reasoning from narratives, the predominant representation has been first-order logic [8, 11]. Recent work on statistical mining of narratives [6, 14] strives to find narrative patterns from large web-corpora. Rishes et al. have proposed an automatic method for converting between the Story Intention Graph (SIG) representation to a natural language generator such as PERSONAGE [13].

The process that Skald undergoes is analogous to that of a human storyteller, in that the system considers and modifies past story examples. However, Skald generates a graph representing a bare plotline as its output, and this representation is insufficient for more rich and complex narratives. Thus far, SIGs have only been applied as an analytical tool on pre-written stories with simple plot structures and character attributes. However, SIGs have the potential to express a richer set of stories when combined with a sufficiently creative generator. Once a narrative is represented in terms of SIGs, we can then transform the story with these SIG representations to result in creative retellings.

4 Translating Generated Plotlines into SIGs

We have developed a system that takes in Skald story data as input and produces SIG encodings. Figure 1 shows a block diagram that details the main steps of the procedure, and the following sections will describe each component of the system in detail.

4.1 Event Ordering

Skald generates a story graph without always indicating the ordering of frames. While not every narrative generation system may require event ordering, we included a module for this purpose so that any story generated by Skald will be told in the proper sequence.

Table 3 An example that demonstrates how frames from Story A are sorted by the EOM.

Sorting Step	Order of Events
1	t1: attack -*intends*- injured, t2: hates -*motivates*- wantinjure, t3: injured -*thwarts*- stayhealthy, t4: wantinjure -*plans*- attack
2	t1: attack -*intends*- injured, t2: injured -*thwarts*- stayhealthy, t3: hates -*motivates*- wantinjure, t4: wantinjure -*plans*- attack
3	t1: hates -*motivates*- wantinjure, t2: wantinjure -*plans*- attack, t3: attack -*intends*- injured, t4: injured -*thwarts*- stayhealthy
4	t1: hates -*motivates*- wantinjure, t2: wantinjure -*plans*- attack, t3: attack -*intends*- injured, t4: injured -*thwarts*- stayhealthy
5	t1: hates -*motivates*- wantinjure, t2: wantinjure -*plans*- attack, t3: attack -*intends*- injured, t4: injured -*thwarts*- stayhealthy

While frames generated by the original Skald system are not ordered in the natural language telling, their implied ordering may be discerned by examining the graph connections between events. We define a *frame pairing* as a set of two frames generated by Skald, wherein one directly connects to the second. For instance, Fafnir attacking Frederick in Story A is connected to his intention to injure him by an *intends* link. In this example, the attacking action intends the injured state, and *attack* and *injured* are a pair.

The Event-Ordering Module (EOM) works as follows: for each frame-consequence pairing, search for the given consequence in the remaining events. If the frame is found, swap the found frame to directly follow the current pairing; then, continue reading through the list. If the frame is not found, move the lines succeeding the current line to the head of the list of frame-consequence pairings; then, begin reading again from the beginning. If not found last, the frame with a consequence matching the final frame is tagged so the module does not check the final two pairings, which should be already sorted.

4.2 Node Construction

In accordance with Elson [4], the Node Constructor (NC) unit categorizes each story element as a Proposition (P), Goal (G), or Belief (B) node. Skald already labels frames as states, goals, and actions, which simplifies the conversion process. Every element of the output graph must then be translated into a discourse relation and annotated with the correct agents, objects, and any other related entities as defined by Elson [4]. Because Beliefs and Goals are frames containing content, they are labeled and filled with one or more Interpretive Proposition (I) relations. In Skald, the affectual impact of a P node or actualized I node is merely implied with frame-consequence pairings and whether goals are achieved. To create a proper SIG encoding, Affectual (A) nodes are created for each character of the story.

Table 4 An example narrative generated by Skald ('Story A'). The story events have been manually ordered and translated into natural text for readability.

Order (t)	Node	Links
1	P: injured(Frederick, False)	*actualizes* (t2)
2	G (Frederick): injured(Frederick, False)	*provides for* A: Frederick
3	G (Fafnir): harm(Fafnir, Frederick)	*provides for* A: Fafnir; *damages* A: Frederick
4	P: attack(Fafnir, Frederick)	*actualizes* (t3)
5	P: injured(Frederick, True)	*ceases* (t2)

4.3 Chain Construction

Once all nodes are established, they must be linked to complete the SIG encoding process. This process is ensured by the Chain Constructor (CC) module, which reviews the given frame-consequence pairings to make decisions about how P and I nodes (including Goals and Beliefs) are linked. For instance, consider the original pairing of 'wantinjure -*plans*-attack' in Story A. In this case, *wantinjure* is classified as a Goal, and *attack* is known to be a P node that takes place in at t=4. Fafnir deciding to attack Frederick, then, at least *attempts to cause* the state of Frederick becoming injured. The attack also intends and results in Frederick becoming injured at t=5, which thwarts his plan to stay healthy. Consequently, a *ceases* link is established between Frederick's goal to stay healthy, and the P node representing the attack in the story. Notably, the previous *attempt to cause* link is changed to become *actualizes*, as Fafnir succeeded in his goal of injuring Frederick.

The system connects each I node to corresponding A nodes by considering the effects of that I on each agent's goals. If a goal is met for an agent when an I node is carried out, a *provides-for* link is established between an agent and that node. Conversely, a *damages* link is created when the current I node thwarts an agent's goal. If any A nodes contain no links by the end of the chain construction process, they are removed from the final graph.

4.4 Output Visualization

At present, our system outputs text that describes a graph structure representing the SIG encodings; Table 4 conveys this information. An example of how this graph would be represented using Story A and Elson's timeline format is shown in Figure 2, while a second story (Story B) is shown in Figure 3.

5 Perspectives and Future Work

By providing Skald with a SIG case library and specifying rules for SIG-based transformations, we can apply the TRAM procedure to the SIGs themselves. For instance, Story A matches the 'Goal (Desire to Harm)' SIG pattern. By instructing Skald to examine the underlying components of the SIG, and searching for similar patterns, the elements of the original story are then adapted for use in a new SIG template. Thus, when transforming Story A, multiple new stories should be produced. For instance, our modified version of Skald could use a GeneralizeLink TRAM template to recognize that the *actualizes* link at t4 can be replaced with an *attempt to cause* link. An *actualizes* link is then created between t4 and a new I node which represents the opposite of the *injures* action ('heals'). Based on the original

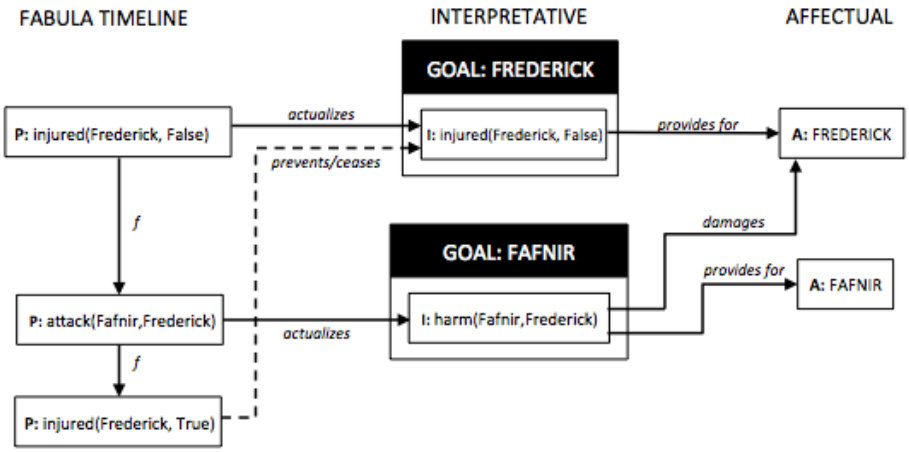

Figure 2 A visual example of the completed SIG encoding for Story A. Story A ultimately follows the 'Goal (Desire to Harm)' SIG pattern.

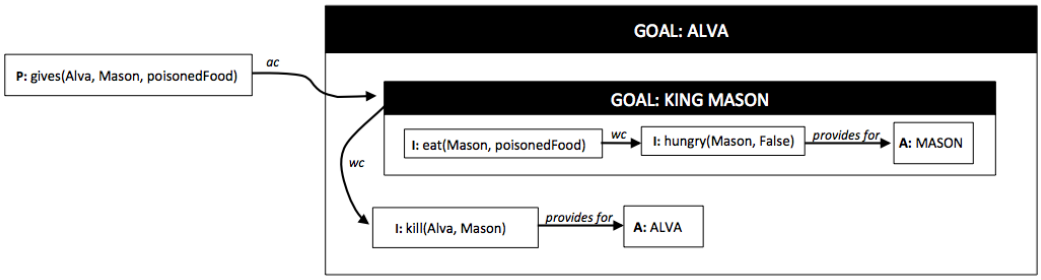

Figure 3 A SIG encoding derived from a second story generated by Skald ("Story B"). Story B includes nested goals and follows the 'Hidden Agenda' pattern. In Story B, a witch named Alva wants to kill King Mason by giving him poisoned food. Mason is hungry, and so accepts the food. Both Alva and Mason's goals are achieved; however, Mason dies by the end of the story.

narrative constraints, the system understands that Frederick being healed is consistent with his goals and thwarts Fafnir's goals, leading to the appropriate connections between the A nodes. The final state, Frederick not being injured, is updated based on the new I node. However, because this state was already a part of the timeline (t1), the final state is removed from the graph, and Frederick's goal by the end of the story is achieved. The resulting story follows the 'Unintended Aid' SIG pattern (Figure 4).

6 Conclusion

We have prepared Skald for improved natural language generation by (1) ordering the frames it produces in graph form, and (2) encoding the story events with story intention graphs. Further, we have extended Skald as a creative system by adding SIGs as a second means of transforming generated stories. Rather than having independent architectures with distinct ways of implementing narrative structure, we can generate more complex stories by working from the SIG specification directly. Output text of other generators may be re-encoded as SIGs, thus enabling comparison between different story generation systems.

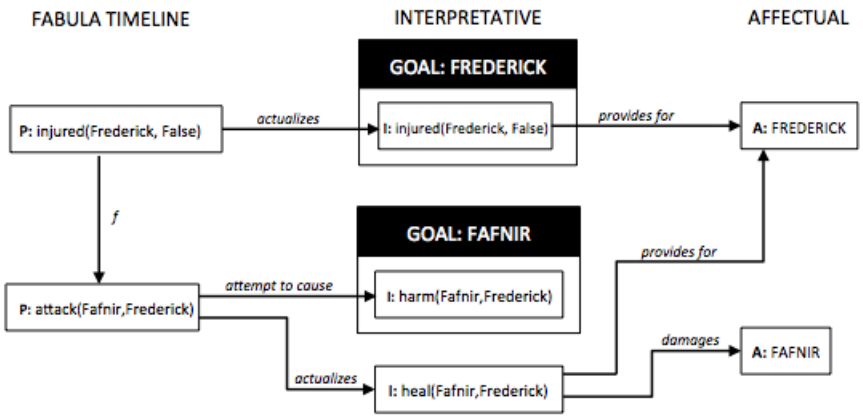

Figure 4 The visual SIG encoding for Story A, when transformed by a modified version of Skald.

The SIG representation, and others like it, enable the expansion of surface realization as an expressive medium. This is true even when the general plots are predictable, implying that stories may be improved even with the same knowledge structures. Future research should work towards quantifying this improvement, as well as to further increase the creative capacity of narrative systems. Future research could also work towards applying the SIG translation process to creative narrative generators beyond Skald, and analyzing variations in the types and diversity of SIG encodings they are able to produce.

Acknowledgements. Sarah Harmon was supported through the BSOE seed funding program. We would also like to thank Brandon Tearse for the development of Skald, Peter Mawhorter for his assistance in reimplementation, and Noah Wardrip-Fruin for his feedback on SIG representation.

References

1 Byung-Chull Bae and R. Michael Young. A use of flashback and foreshadowing for surprise arousal in narrative using a plan-based approach. *Interactive Storytelling*, 4:156–167, 2008.

2 Yun-Gyung Cheong and R. Michael Young. Narrative generation for suspense: Modeling and evaluation. *Interactive Storytelling*, 4:144–155, 2008.

3 Natalie Dehn. Story generation after TALE-SPIN. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, pages 16–18, 1981.

4 David K. Elson. Detecting story analogies from annotations of time, action and agency. In *Proceedings of the LREC 2012 Workshop on Computational Models of Narrative*, Istanbul, Turkey, 2012a.

5 David K. Elson. Dramabank: Annotating agency in narrative discourse. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC 2012)*, Istanbul, Turkey, 2012b.

6 Andrew S. Gordon and Reid Swanson. Identifying personal stories in millions of weblog entries. In *Proceedings of the Third International AAAI Conference on Weblogs and Social Media*, San Jose, California, 2009.

7 Arthur C. Graesser, Kathy L. Lang, and Richard M. Roberts. Question answering in the context of stories. *Journal of Experimental Psychology: General*, 120:254–277, 1991.

8 Jerry R. Hobbs and Andrew S. Gordon. Encoding knowledge of commonsense psychology. In *Proceedings of the 7th International Symposium on Logical Formalizations of Commonsense Reasoning*, pages 107–114, Corfu, Greece, 2005.

9 Sheldon Klein, John F. Aeschlimann, David F. Balsiger, Steve L. Converse, Claudine Court, Mark Foster, Robin Lawo, John D. Oakley, and Joel Smith. Automatic novel writing: A status report. Technical report 186, Computer Science Department, The University of Wisconsin, Madison, 1973.

10 James R. Meehan. Tale-spin, an interactive program that writes stories. In *Proceedings of the fifth International Joint Conference on Artificial Intelligence (IJCAI'77)*, volume 1, pages 91–98, Cambridge, MA, 1977. San Mateo, CA: Morgan Kaufmann.

11 Erik T. Mueller. Modelling space and time in narratives about restaurants. In *Literary and Linguistic Computing*, volume 4, 2006.

12 James Niehaus and R. Michael Young. A computational model of inferencing in narrative. *AAAI Spring Symposium: Intelligent Narrative Technologies II*, 2009.

13 Elena Rishes, Stephanie M. Lukin, David K. Elson, and Marilyn A. Walker. Generating different story tellings from semantic representations of narrative. In *Koenitz, H., Sezen, T.I., Ferri, G., Haahr, M., Sezen, D., C atak, G. (eds.) ICIDS 2013, LNCS*, volume 8230, pages 192–204. Springer, Heidelberg, 2013.

14 Reid Swanson and Arnav Jhala. A crowd-sourced collection of narratives for studying conflict. In *Language Resources and Evaluation Conference Workshop on Computational Models of Narrative (LREC 2012)*, Istanbul, Turkey, 2012.

15 Brandon Tearse. Minstrel Remixed and Skald, 2015. University of California, Santa Cruz, 2012. Web.

16 Brandon Tearse, Peter Mawhorter, Michael Mateas, and Noah Wardrip-Fruin. Skald: Minstrel reconstructed. *IEEE Transactions on Computational Intelligence and AI in Games*, 6:156–165, 2014.

17 Scott Turner. Minstrel: a computer model of creativity and storytelling. Technical Report CSD-920057, Ph.D. Thesis, Computer Science Department, University of California, Los Angeles, CA, 1992.