Towards Controllable Story Generation

Nanyun Peng Marjan Ghazvininejad Jonathan May Kevin Knight
Information Sciences Institute & Computer Science Department
University of Southern California
{npeng,ghazvini,jonmay,knight}@isi.edu

Abstract

We present a general framework of analyzing existing story corpora to generate controllable and creative new stories. The proposed framework needs little manual annotation to achieve controllable story generation. It creates a new interface for humans to interact with computers to generate personalized stories. We apply the framework to build recurrent neural network (RNN)-based generation models to control story ending valence[1] (Egidi and Gerrig, 2009) and storyline. Experiments show that our methods successfully achieve the control and enhance the coherence of stories through introducing storylines. with additional control factors, the generation model gets lower perplexity, and yields more coherent stories that are faithful to the control factors according to human evaluation.

1 Introduction

Storytelling is an important task in natural language generation, which plays a crucial role in the generation of various types of texts, such as novels, movies, and news articles. Automatic story generation efforts started as early as the 1970s with the TALE-SPIN system (Meehan, 1977). Early attempts in this field relied on symbolic planning (Meehan, 1977; Lebowitz, 1987; Turner, 1993; Bringsjord and Ferrucci, 1999; Perez and Sharples, 2001; Riedl and Young, 2010), case-based reasoning (Gervas et al., 2005), or generalizing knowledge from existing stories to assemble new ones (Swanson and Gordon, 2012; Li et al., 2013). In recent years, deep learning models are used to capture higher level structure in stories. Roemmele et al. (2017) use skip-thought vectors (Kiros et al., 2015) to encode sentences, and a Long Short-Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997) to generate stories. Martin et al. (2017) train a recurrent encoder-decoder neural network (Sutskever et al., 2014) to predict the next event in the story.

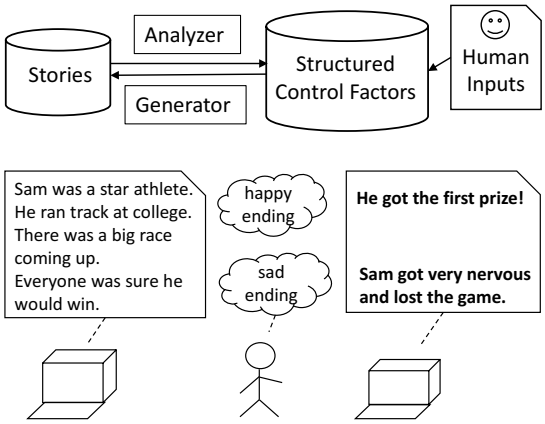

Figure 1: An overview (upper) and an example (lower) of the proposed analyze-to-generate story framework.

Despite significant progress in automatic story generation, there has been less emphasis on *controllability*: having a system takes human inputs and composes stories accordingly. With the recent successes on controllable generation of images (Chen et al., 2016; Siddharth et al., 2017; Lample et al., 2017), dialog responses (Wang et al., 2017), poems (Ghazvininejad et al., 2017), and different styles of text (Hu et al., 2017; Ficler and Goldberg, 2017; Shen et al., 2017; Fu et al., 2017). people would want to control a story generation system to produce interesting and personalized stories.

This paper emphasizes the *controllability* aspect. We propose a completely data-driven approach towards controllable story generation by analyzing the existing story corpora. First, an analyzer extracts control factors from existing stories, and then a generator learns to generate stories according to the control factors. This creates an excellent interface for humans to interact: the generator can take human-supplied control factors to generate stories that reflect a user's intent. Fig-

[1] Happy or sad endings.

Proceedings of the First Workshop on Storytelling, pages 43–49
New Orleans, Louisiana, June 5, 2018.

ure 1 gives the overview (upper) and an example (lower) of the framework. The instantiations of the analyzer and the generator are flexible and can be easily applied to different scenarios. We explore two control factors: (1) ending valence (happy or sad ending) and (2) storyline keywords. We use supervised classifiers and rule-based keyword extractors for analysis, and conditional RNNs for generation.

The contributions of the paper are two-fold:

1. We propose a general framework enabling interactive story generation by analyzing existing story corpora.
2. We apply the framework to control story ending valence and storyline, and show that with these additional control factors, our models generate stories that are both more coherent and more faithful to human inputs.

2 Controllable Story Generation

As a pilot study, we explore the control of 1) ending valence, which is an abstract, style-level element of stories, and 2) storyline, which is a more concrete, content-level concept for stories.

2.1 Ending Valence Control

Prior work has explored manipulating emotion in interactive storytelling (Cavazza et al., 2009). For simplicity, we refine our scope to manipulating the ending valence for controllable story generation. We categorize ending valence into *happyEnding, sadEnding*, or *cannotTell*.

Analyzer. The analyzer for the ending valence control is a *classifier* that labels each story as *happyEnding, sadEnding*, or *cannotTell*. Formally, given a story corpus $X = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$ with N stories, the ending valence analyzer is a function f^v that maps each story $\mathbf{x}_i$ to a label l_i:

$$l_i = f^v(\mathbf{x}_i),$$

where i indexes instances. Since there is no prior work on analyzing story ending valence, we build our own analyzer by collecting some annotations for story ending valence from Amazon Mechanical Turk (AMT) and building a supervised classifier. We employ an LSTM-based logistic regression classifier as it learns feature representations that capture long-term dependencies between the words, and has been shown efficient in text classification tasks (Tang et al., 2015).

Specifically, we use a bidirectional-LSTM to encode an input story into a sequence of vector representations $\mathbf{h}_i = \{h_{i,1}, h_{i,2}, \cdots, h_{i,T}\}$, where $\mathbf{h_i} = BiLSTM(\mathbf{x}_i) = [\overrightarrow{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_i]$, T denotes the story length, $[,:,]$ denotes element-wise concatenation. $\overrightarrow{\mathbf{h}}_i$ and $\overleftarrow{\mathbf{h}}_i$ are sequences of vectors computed by a forward and a backward LSTM. an LSTM-cell is applied at each step to complete the following computations:

$$\begin{pmatrix} i_{i,t} \\ f_{i,t} \\ o_{i,t} \\ c_{i,t} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} U \begin{pmatrix} E^w x_{i,t} \\ h_{i,t-1} \end{pmatrix} \quad (1a)$$

$$\tilde{c}_{i,t} = f_{i,t} \odot \tilde{c}_{i,t-1} + i_{i,t} \odot c_{i,t} \quad (1b)$$

$$h_{i,t} = LSTM\text{-}cell(x_{i,t}, h_{i,t-1}) \quad (1c)$$

$$= o_{i,t} \odot \tanh(\tilde{c}_{i,t}) \quad (1d)$$

$i_{i,t}, f_{i,t}, o_{i,t}, c_{i,t}$ are the input, forget, output gates, and a contemporary central memory that control the information flow of the previous contexts and the current input. σ and $\tanh$ denotes element-wise sigmoid and tanh function. E^w is an embedding matrix that maps an input word $x_{i,t}$ to a x-dimensional vector. Each $h_{i,t}$ is a d-dimensional vector that can be viewed as the contextual representation of word $x_{i,t}$.

To obtain the sentence representation, we take a max pooling over the sentence, where for each dimension j of the vector $\hat{h}_i$, we have:

$$\hat{h}_i^j = \max_{t \in [1,\ldots,T]} h_{i,t}^j, \; j = 1, ..., d. \quad (2)$$

The final classifier is defined as:

$$f^v(\mathbf{x}_i) = g(W\hat{h}_i + b), \quad (3)$$

where $g()$ is the softmax function, W and b are model parameters that are jointly learned with the BiLSTM parameters.

Generator. The generator for the ending valence-controlled story generation is a conditional language model, where the probability of generating each word is denoted as $p(w_t|w_1^{t-1}, l; \theta)$; l represents the ending valence label and θ represents model parameters. We learn valence embeddings for the ending valence labels to facilitate the computation. Formally, we learn an embedding matrix E^l to map each label l^k into a vector:

$$e_l^k = E^l[l^k],$$

where E^l is a $m \times p$ matrix that maps each label (p of them) into a m-dimensional vector. The ending valence embeddings dimension are made the same as the word embedding dimension for simplicity.

We add the ending valence as follows:

$$p(w_t|w_1^{t-1}, l; \theta) = \begin{cases} g(V\mathcal{F}(e_l, \mathcal{F}(w_{t-1}, h_{t-1}))), t = s \\ g(V\mathcal{F}(w_{t-1}, h_{t-1})), t = others \end{cases} \quad (4)$$

where s denotes the position right before the ending sentence, $g()$ is the softmax function, $\mathcal{F}$ means the computations of an LSTM-cell, and V denotes parameters that perform a linear transformation. We treat the ending valence as an additional input to the story. The valence embeddings are jointly learned with other model parameters.

2.2 Storyline Control

Li et al. (2013) introduced plot graphs which contain events and their relations to represent storyline. Although the representation is rich, these plot graphs are hard to define and curate without highly specialized knowledge. In this pilot study, we follow what Yan (2016) did for poetry, to use *a sequence of words* as the storyline. We further confine the words to appear in the original story.

Analyzer. The analyzer for storyline control is an *extractor* that extracts a sequence of words $\mathbf{k}_i = \{k_{i,1}, k_{i,2}, \ldots, k_{i,r}\}$ from each story $\mathbf{x}_i$. The k_is are ordered according to their order in the story. We adapt the RAKE algorithm (Rose et al., 2010) for keyword extraction, which builds document graphs and weights the importance of each word combining several word-level and graph-level criteria. We extract the most important word from each sentence as the storyline.

Generator. The generator for storyline-controlled generation is also a conditional language model. Specifically, we employ the seq2seq model with attention (Bahdanau et al., 2014) implemented in OpenNMT (Klein et al., 2017). Specifically, the storyline words are encoded into vectors by a BiLSTM: $\mathbf{h}^k = BiLSTM(\mathbf{k}) = [\overrightarrow{\mathbf{h}}^k; \overleftarrow{\mathbf{h}}^k]$, and the decoder generate each word according to the probability:

$$p(w_t|w_1^{t-1}, \mathbf{h}^k; \theta) = g(V^l s_t) \quad (5a)$$

$$s_t = \mathcal{F}^{\text{att}}(w_{t-1}, s_{t-1}, c_t) \quad (5b)$$

$$c_t = \sum_{j=1}^{r} \alpha_{tj} h_j^k \quad (5c)$$

$$\alpha_{tj} = \frac{\exp(a(s_{t-1}, h_j^k))}{\sum_{p=1}^{r} \exp(a(s_{t-1}, h_p^k))} \quad (5d)$$

Agreement experiment	Cases	Agreement
Researcher vs. Researcher	150	83%
Turkers vs. Researcher	150	78%
Classifier vs. Turkers	3980	69%
Always *happyEnding*	3980	58%

Table 1: Annotation agreement for labeling story ending valence. Labels are *happyEnding*, *sadEnding*, or *cannotTell*. The automatic classifier trained on 3980 turker annotated stories achieved much better results than the majority baseline on 5-fold cross-validation.

Method	PPL
uncontrolled model	24.63
Storyline controlled	**18.36**

Table 2: Perplexities on the ROCstories development data. When storylines are given, the controlled models achieve lower perplexity than the uncontrolled one.

$g()$ again denotes the softmax function, and V^l denotes parameters that perform a linear transformation. $\mathcal{F}^{\text{att}}()$ in Equation 5b denotes the computations of an LSTM-cell with attention mechanism, where the context vector c_t is computed by an weighted summation of the storyline words vectors as in Equation 5c, and the weights are computed from some alignment function $a()$ as in Equation 5d.

3 Experimental Setup

We conduct experiments on the ROCstories dataset (Mostafazadeh et al., 2016), which consists of 98,162 five-line stories for training, and 1871 stories each for the development and test sets. We treat the first four sentences of each story as the body and the last sentence as the ending. We build analyzers to annotate the ending valence and the storyline for every story, and train the two controlled generators with 98,162 annotated stories.

3.1 Ending Valence Annotation

We conduct a three-stage data collection procedure to gather ending valence annotations and train a classifier to analyze the whole corpora. We classify all the stories into *happyEnding*, *sadEnding*, or *cannotTell*. Table 1 summarizes the results.

In the first stage, two researchers annotate 150 stories to gauge the feasibility of the task. It is nontrivial, as the agreement between the two researchers is only 83%, mainly because of the *cannotTell* case[2]. The second stage collects larger-

[2]The inter-annotator agreement is 95% if we exclude the instances that at least one person chose *cannotTell*.

Methods	Ending Valence				Storyline			
	Faithfulness		Coherence		Faithfulness		Coherence	
	avg score	% win	avg score	% win	avg score	% win	avg score	% win
Retrieve Human	3.20	19.6	2.89	16.4	**3.47**	**42.4**	3.44	17.6
Uncontrolled	3.26	27.8	**3.54**	**41.8**	2.82	20.8	3.54	29.2
Controlled-Generated	**3.44**	**52.6**	3.37	**41.8**	3.08	36.8	**3.74**	**53.2**

Table 3: Human evaluation for the ending valence (left) and storyline (right) controlled generation. Scores range in [1,5]. Three stories (one from each method) are grouped together so that people can give comparative scores. Faithfulness survey asks people to rate whether the generated stories reflect the given control factors. Coherence asks people to rate the coherence of the stories without considering the control factors. % win measures how often the generated result by one method is rated higher than others, excluding the instances that tie on the highest score.

scale annotations from AMT. We gather 3980 annotated stories with the turker-researcher agreement at 78%. A classifier as described in Section 2.1 is then trained to analyze the whole ROC-stories corpora. Using 5-fold cross-validation, we estimate the accuracy of the classifier to be 69%[3], which, while not terribly impressive, is an 11% improvement over the majority baseline (*happyEnding*). Considering the low inter-annotator agreement on this problem, we consider this a decent analyzer.

4 Experimental Results

We compare the controlled generation under our proposed framework with the uncontrolled generation. We design the experiments to answer the following research questions:

1. How does the controlled generation framework affect the generation quantitatively?
2. Does the proposed framework enables controls to the stories while maintaining the coherence of the stories?

To answer the former question, we design automatic evaluations that measure the perplexity of the models given appropriate and inappropriate controls. For the latter question, we design human evaluations to compare the generated stories from controlled and uncontrolled versions in terms of the document-level coherence and the faithfulness to the control factors.

4.1 Automatic Evaluation

The advantages of the automatic evaluation is that it can be conducted at scale and gives panoramic views of the systems. We compute the perplexities of different models on the ROCstories development dataset. Table 2 shows the results for the storyline experiments. With the additional storyline information, it is easier for the generation model to guess what will happen next in a story, thus yield lower perplexities. We conduct the same experiments for ending valence controlled generation and observe the same. However, since ending valence is only one bit of information, the perplexity difference is only 0.8.

4.2 Human Evaluation

We conduct a human evaluation with 1000 story groups for each setting. Each group consists of stories from: (1) the uncontrolled LSTM generation model, (2) controlled generation with our framework, and (3) a contrastive method which retrieves and re-ranks existing sentences in the training data. Users are asked to rate the three stories on a 1-5 scale with respect to faithfulness (whether stories reflect the control factor), and coherence. All the evaluations are conducted on Amazon Mechanic Turk. We compute the average score and percentage win of each method. Table 3 summarizes the results.

Ending Valence For the ending valence control, we supply each system with the first 4 sentences from ROCStories test set and an ending valence randomly assigned by a human. The systems generate endings[4]. We only let the systems generate *happyEnding* or *sadEnding* stories, with the ratio around 1:1. Faithfulness is defined as whether the generated stories reflect the given ending valence.

The contrastive method retrieves existing happy or sad endings from the training data instead of generating new sentences. Specifically, we gather all the stories that are annotated with *happyEndings* from the 3980 annotated stories in one set, and all the *sadEndings* in another set. When the given ending valence is *happyEnding*, the sys-

[3]We included the *cannotTell* cases and conducted a 3-class classification.

[4]The uncontrolled LSTM generation model has no way to take the ending valence input.

Story Body	Setting	Ending
sarah had been dreaming of visiting europe for years. she had finally saved enough for the trip. she landed in spain and traveled east across the continent. she did n't like how different everything was.	Human	sarah decided she preferred home over europe.
	No-Valence-Control	sarah was so happy that she had a great time.
	Happy-Generated	sarah was happy to go home.
	Sad-Generated	sarah was very disappointed.
	Happy-Retrieved	they both had a lot of fun.
	Sad-Retrieved	I had to go to the hospital.
samuel loved reading old science fiction stories. he collected books by hg wells and jules verne. his favorite was hg wells. his father gave him a copy of the island of dr moreau.	Human	he loved it!
	No-Valence-Control	he was so happy to see the book.
	Happy-Generated	he was very happy to see the new book.
	Sad-Generated	he was so sad that he couldn't find his book.
	Happy-Retrieved	by the end of the day, he was glad he had gone!
	Sad-Retrieved	bob got very sick and had to go to the hospital.

Table 4: Samples of endings generated by given the first four sentences of a story and an ending valence. The uncontrolled version cannot take the designated ending valence into account, while both the controlled versions reflect the correct valence.

Users' Storyline	Generated Story
john angry meeting park happy	john was a very angry person. he was angry at his job. john decided to go to the park. john went to the park. john was very happy with his new job.
john angry meeting park wolf	john was very angry. he had a meeting at the park. he went to the park. he saw a wolf. john was very angry.
jennifer angry meeting park happy	jennifer was very angry. she had a meeting at the park. jennifer was not happy. she went to the park. jennifer was happy to be meeting her friend.
jennifer angry meeting park wolf scared	jennifer was very angry. she had a meeting at the park. she went to the park. she saw a wolf. jennifer was very angry.

Table 5: Samples of stories generated from user-given storyline words. The generated stories usually contains the words in the storyline. Changing one word in the storyline can result in another creative story.

tem scores all the sentences from the *happenEnding* set by combining it with the given 4 sentences, and using a trained uncontrolled generation model to compute the likelihood. This chooses the most coherent *happyEnding* for the given story. Similarly for the *sadEnding* stories. Table 3 shows that the proposed analyze-to-generate framework ("Controlled-Generated") achieves the highest faithfulness score while retaining similar coherence as the uncontrolled one.

Storyline For storyline control, we supply each system with 5 words as a storyline. The systems generate stories accordingly. Storyline words are extracted from the ROCstories test set. The uncontrolled generation model cannot take this input; it generates random stories. Faithfulness is defined as whether the generated stories follow the given storyline.

The contrastive method retrieves human written sentences in the training data to compose stories. Specifically, it follows the given storyline words order to retrieve sentences from the training data. The trained uncontrolled generation model scores each sentence based on existing previous sentences and choose the highest scoring sentence for each word in the storyline. If a word in the storyline has never appeared in the training data, we simply skip it.

As shown in Table 3, the contrastive method achieves the highest faithfulness, probably because it guarantees the words in the storyline appear in the stories while the other systems cannot. However, the coherence of the contrastive method is lowest, because it is constrained by the existing sentences in the training data. Although an uncontrolled generation model is employed to encourage document-level coherence, the available choices are restricted. Our method achieves the

best coherence and higher faithfulness score than the uncontrolled version.

4.3 Generation Samples

Table 4 shows two examples of the ending valence controlled generation. The uncontrolled model "No-Valence-Control" can generate coherent endings; However, it cannot alter the ending valence. On the other hand, the two controlled models can generate different endings based on different ending valence. The contrastive retrieval method, restricted by the existing *happyEnding* and *sadEnding* in the training data, obtains endings that are not coherent with the whole story.

Table 5 demonstrates some examples from the storyline controlled generation. The storyline words are user supplied. We can see that this provides fun interactions: changing one word in the storyline can result in a creative new story.

5 Conclusion

We proposed an analyze-to-generate framework that enables controllable story generation. The framework is generally applicable for many control factors. In this paper, two instantiations of the framework are explored to control the ending valence and the storyline of stories. Experiments show that our framework enables human controls while achieving better coherence than an uncontrolled generation models. In the future, we will explore other control factors and better controllable generation models to adding the control factors into the generated stories. The current analyze-to-generate framework is done in a pipeline fashion. We also plan to explore the joint training of the analyzer and the generator to improve the quality of both.

Acknowledgements

We thank the anonymous reviewers for the useful comments and the suggestion for the right terminology of "ending valence". This work is supported by Contract W911NF-15-1-0543 with the US Defense Advanced Research Projects Agency (DARPA).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Selmer Bringsjord and David Ferrucci. 1999. *Artificial intelligence and literary creativity: Inside the mind of brutus, a storytelling machine*. Psychology Press.

Marc Cavazza, David Pizzi, Fred Charles, Thurid Vogt, and Elisabeth André. 2009. Emotional input for character-based interactive storytelling. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, pages 313–320.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Proceedings of Advances in Neural Information Processing Systems*.

Giovanna Egidi and Richard J Gerrig. 2009. How valence affects language processing: Negativity bias and mood congruence in narrative comprehension. *Memory & cognition* 37(5):547–555.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. In *Proceedings of the Workshop on Stylistic Variation*. pages 94–104.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2017. Style transfer in text: Exploration and evaluation. In *Proceedings of The 32nd AAAI Conference on Artificial Intelligence*.

Pablo Gervas, Belen Diaz-Agudo, Federico Peinado, and Raquel Hervas. 2005. Story plot generation based on CBR. *Knowledge-Based Systems* 18(4):235–242.

Marjan Ghazvininejad, Xing Shi, Jay Priyadarshi, and Kevin Knight. 2017. Hafez: an interactive poetry generation system. *Proceedings of 55th Annual Meeting of the Association for Computational Linguistics, System Demonstrations* pages 43–48.

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *Proceedings of the International Conference on Machine Learning*. pages 1587–1596.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Proceedings of Advances in neural information processing systems*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *ACL*.

Guillaume Lample, Neil Zeghidour, Nicolas Usunier, Antoine Bordes, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Fader networks: Manipulating images by sliding attributes. In *Proceedings of 31st Conference on Neural Information Processing Systems*.

Michael Lebowitz. 1987. Planning stories. In *Proceedings of the cognitive science society, Hillsdale*.

Boyang Li, Stephen Lee-Urban, George Johnston, and Mark Riedl. 2013. Story generation with crowdsourced plot graphs. In *Proceedings of The 28th AAAI Conference on Artificial Intelligence*.

Lara Martin, Prithviraj Ammanabrolu, William Hancock, Shruti Singh, Brent Harrison, and Mark Riedl. 2017. Event representations for automated story generation with deep neural nets. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

James Meehan. 1977. TALE-SPIN, an interactive program that writes stories. In *Proceedings of The International Joint Conferences on Artificial Intelligence Organization*.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Rafael Perez and Mike Sharples. 2001. MEXICA: A computer model of a cognitive account of creative writing. *Journal of Experimental & Theoretical Artificial Intelligence* 13(2):119–139.

Mark Riedl and Robert Young. 2010. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research* 39:217–268.

Melissa Roemmele, Sosuke Kobayashi, Naoya Inoue, and Andrew M Gordon. 2017. An RNN-based binary classifier for the story cloze test. *LSDSem 2017* page 74.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text Mining: Applications and Theory* pages 1–20.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Proceedings of Advances in neural information processing systems*.

N Siddharth, Brooks Paige, Van de Meent, Alban Desmaison, Frank Wood, Noah D Goodman, Pushmeet Kohli, Philip HS Torr, et al. 2017. Learning disentangled representations with semi-supervised deep generative models. In *Proceedings of Advances in Neural Information Processing Systems*.

Ilya Sutskever, Oriol Vinyals, and Quoc Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of Advances in neural information processing systems*.

Reid Swanson and Andrew Gordon. 2012. Say anything: Using textual case-based reasoning to enable open-domain interactive storytelling. *ACM Transactions on Interactive Intelligent Systems* 2(3):16.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. pages 1422–1432.

Scott R. Turner. 1993. *Minstrel: A Computer Model of Creativity and Storytelling*. Ph.D. thesis, Los Angeles, CA, USA. UMI Order no. GAX93-19933.

Di Wang, Nebojsa Jojic, Chris Brockett, and Eric Nyberg. 2017. Steering output style and topic in neural response generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. pages 2130–2140.

Rui Yan. 2016. i, poet: Automatic poetry composition through recurrent neural networks with iterative polishing schema. In *Proceedings of The International Joint Conferences on Artificial Intelligence Organization*. pages 2238–2244.