# Toward Automated Story Generation with Markov Chain Monte Carlo Methods and Deep Neural Networks

**Brent Harrison, Christopher Purdy, and Mark O. Riedl**
School of Interactive Computing, Georgia Institute of Technology
Atlanta, Georgia, USA
{bharrison6, cpurdy3, riedl}@gatech.edu

## Abstract

In this paper, we introduce an approach to automated story generation using Markov Chain Monte Carlo (MCMC) sampling. This approach uses a sampling algorithm based on Metropolis-Hastings to generate a probability distribution which can be used to generate stories via random sampling that adhere to criteria learned by recurrent neural networks. We show the applicability of our technique through a case study where we generate novel stories using an acceptance criteria learned from a set of movie plots taken from Wikipedia. This study shows that stories generated using this approach adhere to this criteria 85%-86% of the time.

## Introduction

*Automated story generation* is the use of artificial intelligence to craft novel, fictional story content. Automated story generation systems have achieved the greatest successes in fictional worlds for which there are a finite, enumerated set of actions, characters, objects, and settings. Examples include computer game environments (Porteous and Cavazza 2009; Li and Riedl 2010) and micro-worlds based on fairy tales (Meehan 1977; Turner 1994; Pérez y Pérez and Sharples 2001; Riedl and Young 2010; Martens et al. 2014; Farrell and Ware 2016) or soap operas (Lebowitz 1987). In these worlds, a domain author first constructs a model of the fictional world. This domain model could be logical operators, an ontology, etc. A story generation algorithm then reasons about the model to produce a sequence of actions performed by characters. These systems are, by their nature, locked in to one specific domain until the domain model is changed.

*Open story generation* tackles the challenge of generating and telling stories in any conceivable domain without re-learning or retraining the domain model. That is, a user may request a story about any topic he or she can think of and the automated story generation system will be able to respond. There are two primary challenges of open story generation: (1) automatically acquiring a model of story progression, and (2) guiding the story progression progress in the face of uncertainty.

Recently, *recurrent neural networks* (RNNs) have been proposed as a means of automated story generation (Roemmele 2016; Martin et al. 2017; Khalifa, Barros, and Togelius 2017). A recurrent neural network can be trained on a corpus of natural language stories to infer the probability of a character, word, or sentence given one or prior characters, words, or sentences. Stories can then be generated by sampling from this distribution. That is, the problem of story generation with neural networks is cast as a prediction problem. Training the RNN on a text corpus containing stories from many domains and topics can yield a path toward open story generation.

While recurrent neural networks have shown promise in turn-taking dialogue, where one—or a few—natural language utterances follow a human utterance, RNNs have demonstrated difficulty maintaining a coherent progression for more than a few turns. This is especially true when the neural network is trained on a diverse story corpus. One of the reasons for this is that a RNN model trained on text is a *language model*: it models the probability of tokens (characters, words, etc.) being produced given the observation of prior tokens. The act of story writing is better described as a deliberative, planning process (Dehn 1981; Sharples 1999) than a process of sampling conditioned on observations.

In this work, we introduce an approach to open story generation using Markov Chain Monte Carlo (MCMC) search trained on a diverse story corpus. Unlike a neural language model approach to story generation which generates stories by learning a distribution over observations, MCMC methods attempt to approximate a posterior distribution of an unknown function by performing simulations. In this case, the unknown distribution we want to learn is that of a hypothetical storyteller that tells *coherent* stories in which certain events come to pass.

Our contributions are as follows:

- We introduce a MCMC algorithm for open story generation.
- We introduce a technique for using deep neural networks to guide the MCMC search toward producing stories that meet genre expectations.
- We perform a case study in which we use this algorithm for generating stories based on movie plots from Wikipedia, resulting in successfully guided stories 85%-86% of the time.

The remainder of the paper is organized as follows. We will first discuss related research in this area as well as background knowledge on Markov Chain Monte Carlo sampling. Next, we introduce our approach for using MCMC sampling for open story generation. We then discuss a case study we performed which shows how this approach can be used to generate stories using naturally-occurring story corpora. We conclude with a discussion of results, limitations, and opportunities for future work.

## Related Work

Automated Story Generation has been a research problem of interest since nearly the inception of artificial intelligence. Closed world story generation techniques include symbolic and logical planning (Meehan 1977; Lebowitz 1987; Cavazza, Charles, and Mead 2002; Porteous and Cavazza 2009; Pérez y Pérez and Sharples 2001; Riedl and Young 2010; Martens et al. 2014; Farrell and Ware 2016) and case-based reasoning (Turner 1994; Gervás et al. 2005). Symbolic planning systems rely on predicate domain models that state the characters, objects, and actions that can be performed. Case-based reasoning systems require a case base—a database of examples. While this case base can be acquired, most case-based story generation systems to date require stories to be represented according to a preexisting ontology.

Recently, machine learning has been used to attempt to learn story domain models or to identify segments of story content available in an existing repository to assemble stories. The *SayAnthing* system (Swanson and Gordon 2012) uses *textual* case-based reasoning to identify relevant existing story content in online blogs. Unlike earlier case-based approaches, textual case-based reasoning does not require the case base to be represented according to an existing ontology. *SayAnything* is an interactive system in which a human storyteller and a computer take turns. The system searches online blog posts for a sentence similar to that written by a human and then retrieves the next sentence to continue the story. However, the system tends to quickly lose the coherence of the story and requires human intervention to keep it on track.

The *Scheherazade* system (Li et al. 2013) uses a crowdsourced corpus of example stories to learn a domain model from which to generate novel stories. While Scheherazade is an approach to open story generation that does not suffer from coherence issues, every new domain or topic requires a new corpus to be crowdsourced and a new model to be learned. The story representation used by Scheherazade makes it difficult to combine or generalize these models.

Recurrent neural networks can theoretically learn to predict the probability of the next character, word, or sentence in a story. By implementing a softmax layer over outputs, stories can be generated by sampling from the output of a recurrent neural network trained on story text data. Roemmele and Gordon (Roemmele 2016) use a *Long Short-Term Memory* (LSTM) network (Hochreiter and Schmidhuber 1997) to generate stories. Khalifa et al. (Khalifa, Barros, and Togelius 2017) argue that stories are better generated using recurrent neural networks trained on highly specialized textual corpora, such as the body of works from a single, prolific author. However, such a technique is not capable of open story generation as such a network would be too specialized to generate stories in any conceivable domain.

Martin et al. (Martin et al. 2017) showed that they could increase the predictive power of a sequence-to-sequence recurrent neural network (Sutskever, Vinyals, and Le 2014) for story generation if one reduces sentences in a natural language corpus to a specialized event representation containing only the subject, verb, object, and a modifier token. They break the problem of open story generation into two problems: (1) generating a successor event, and (2) translating events back into natural language so that the generated stories are human-readable. In our work, we adopt Martin et al.'s event representation and their decomposition of story generation into successor generation and natural language generation.

## Markov Chain Monte Carlo Simulation

The technique introduced in this paper is based on the Metropolis-Hastings algorithm (Chib and Greenberg 1995) for generating samples from a distribution that is either unknown or otherwise difficult to sample from. This is done by, first, starting with an initial distribution, $X$. A candidate sample, $x \sim X^t$, is drawn from this distribution on iteration $t$ and is either accepted or rejected according to some criteria. Typically, this acceptance criteria is a ratio based on this candidate sample, $x$, compared to the previous accepted sample, $y \sim X^{t-1}$. Given a value function $f(x)$, the probability of acceptance $\alpha = min(1, \frac{f(x)}{f(y)})$. If the candidate sample is accepted, then the distribution $X^t$ is updated with this new information, becoming $X^{t+1}$. If the candidate sample is not accepted, then $X$ remains unchanged. After generating samples for some number of initial iterations, $T$, often referred to as a *burn-in* period, the distribution $X^T$ is considered to be an approximation for the true, underlying distribution that we wished to sample from.

This places a great deal of importance on how the value function $f(x)$ is chosen, as it determines how the sample distribution will change over time. In this paper, we will use this function to help guide story generation using MCMC sampling.

## MCMC for Story Generation

The goal of this approach is to construct a distribution over sentences that can be used to create stories that fit a predefined pattern. It is difficult to directly sample from this distribution since the space of all possible stories, even when using a fixed vocabulary, is very large. Even performing MCMC sampling over a distribution of words to create stories can be difficult because it is likely to result in stories that fail to maintain context or any coherent plot. In this section we will first discuss three simplifications that make this problem tractable: constructing a conditional word distribution for sampling, using an event representation, and implementing vocabulary restraints. We will then discuss, in

greater detail, the acceptance criteria that we use to guide the MCMC sampling process.

### Conditional Word Distribution

In order to use this technique, we first need to define a sampling distribution, $X^0$. The ultimate goal of this work is to use this distribution to construct stories, thus one option is for this distribution to be over all possible stories. This is infeasible as the space of all possible stories is impossible to enumerate. To simplify this we instead define a conditional distribution over individual words that can be used to sample stories. This greatly reduces the complexity of the sampling distribution; however, this distribution contains minimal history that can be exploited to generate the story. Specifically, the words that are sampled from this distribution are conditioned on the previous word sampled, meaning that this is a distribution over bigrams of words that can appear in the story. Limiting the distribution to bigrams is acceptable in this case as this is the *initial* distribution; we will use the acceptance criteria to help encode history into this distribution. This will be discussed in more detail later in the paper.

### Event Representation

In this work, we consider a story to be a Markov chain where each element of the chain is sampled from a distribution, $X$, over possible bigrams that can be in the story. To simplify the problem, we use the simplification trick from Martin et al. (2017) and use *events* to represent sentences in the story. In this work, each event consists of the following elements: $\langle subject, verb, object, modifier \rangle$. That is, every sentence in the original story corpus is reduced to a 4-word sequence of this form. Sentences with more than one verb or more than one subject are broken up into multiple events. The event representation drastically reduces the amount of words that need to be generated in order to produce a story. Instead of having to sample the bigram distribution to generate each word in each sentence of the story, we simply have to sample four words for each sentence in the story.

In practice, however, this event representation combined with the conditional distribution used for sampling can cause problems. If each word in the sampling distribution is only conditioned on the word that comes before it, then it is very difficult for samples drawn from this distribution to adhere to this event structure. This is because a simple conditional distribution cannot effectively encode position information for a 4-tuple. For instance, if the first word generated was a noun it is possible to draw a verb from the distribution (in which case the preceding noun was the subject of the event tuple), but it is also possible to draw another noun from the distribution (in which case the preceding noun was likely the object of an event tuple because the modifier slot can also contain a noun). This ambiguity makes it impossible to use this conditional distribution to reliably reconstruct event structure. To solve this, we add position constraints to the distribution.

### Position Constraints

In order to solve the ordering issue, we alter the sampling distribution such that words are conditioned on both the previous word as well as that word's position. That, is we sample words from the distribution $P(x|X, i)$ where $i$ is the word's position in the story. This effectively encodes event structure into the distribution, resolving any potential ambiguities. Consider the example used in the previous section. If given the information that a noun was previously generated, it is ambiguous as to what can possibly come next. If the positional knowledge that the noun was generated in the first (i.e., $i = 0$) position is also used to condition the next word, then we know that a verb must be generated in the $i = 1$ position. This additional position information also encodes vocabulary restrictions in each position, further improving the quality of generated events using random sampling. In addition to improving the quality of generated events, this additional information improves story quality as it helps encode long-term story trajectory.

## Acceptance Criteria

The acceptance criteria, $f(x)$, forces the distribution $X$ to change toward the final, *a priori* unknown distribution. As each sample $x$ is drawn from the current $X^t$ distribution, $f(x)$ provides a score. If the score is greater than the sample drawn from the previous iteration's distribution $X^{t-1}$, then $X^t$ is updated. Otherwise, the distribution is unchanged. We implement two acceptance criteria, described in the next sections, which are then multiplied together.

### Event Succession

Our first acceptance criterion implements a model of *event succession*. Certain events are more likely to occur after other events. Since the original word bigram distribution cannot capture long-term word dependencies, we score sequences of eight words (two events) according to whether they are supported by the corpus.

We use a sequence-to-sequence neural network (Sutskever, Vinyals, and Le 2014) to learn a model of event succession. A sequence-to-sequence neural network comprises two neural networks trained together in an end-to-end process. The first network is an *encoder*, which learns a latent representation of the input sequence. The second network is a *decoder*, which learns to generate a sequence from the latent representation. The network learns to compute the probability of an output sequence $y_1, ..., y_{T'}$, given an input sequence $x_1, ..., x_T$. That is:

$$p(y_1, ..., y_{T'}|x_1, ..., x_T) = p(y_t|v, y_1, ..., y_{t1}) \quad (1)$$

where $v$ is the latent vector learned by the network.

We train the sequence-to-sequence network by providing pairs of inputs and outputs such that the input is event $t$ and the output is event $t + 1$, as shown in Figure 1. The event data is the same story corpus represented as events as used by the MCMC simulation.

Once the event succession model is trained, we send every group of eight words—two events—from the MCMC sampled story through the sequence-to-sequence neural network to compute the log-probability that the first event is succeeded by the second event. These individual log-probabilities are then summed in order to obtain the cumulative log-probability of the sampled story.

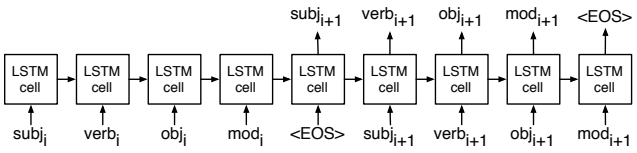


Figure 1: Sequence-to-sequence neural network.

## Long-Range Event Relationships

The event succession model above will cause MCMC to learn a posterior distribution similar to that learned by the sequence-to-sequence network. That is, the result would be no different than generating stories using an LSTM trained on a story corpus, e.g., Martin et al. (2017). However, events can have long-distance relationships that cannot easily be captured by an event succession model. For example, we might want stories with marriages to first have one character fall in love with another character. This is a standard pattern of character behavior in stories that is not easily captured by the event succession model; "fall in love" and "get married" may not occur near to each other in a story.

One of the advantages of generating stories with MCMC is that it generates arbitrarily long sequences, providing some ability to perform lookahead to future events. Our second acceptance criterion attempts to push the system's posterior distribution toward learning long-range relationships between events. Our acceptance criteria takes a $k$-length sequence of verbs and scores a generated story according to whether the $k$ verbs appear in the specified order *without* consideration for adjacency.

Any sequence of $k$ verbs can be provided to this acceptance criteria based on human intuition. The list of verbs is used as an acceptance criteria as follows. We scan through the Markov chain of events sampled from the MCMC's current distribution. For each of the verbs $v_k$ found in word position $i$, we check that $(i_k \mod 4) = 1$ (i.e., it is the second word in an event) and that $i_k > i_{k-1}$. If this is true for each verb provided, then this score for the sample is set to one. If it is not true, then this score for the sample is set to zero. Since we multiply the long-range relationship score with the event succession score order to determine overall sample acceptance, this means that each accepted sample must adhere to the desired verb order. This ensures that the bigram distribution learned will be skewed towards stories that adhere to this criteria.

## Skipping Recurrent Neural Networks

Ideally, we would like to guide the MCMC algorithm with long-range event dependencies that are not reliant on human intuition. We train a second neural network to generate *plot summaries* for a corpus of stories. A plot summary is a list of the $k$ most important events in a story. We use a specialized neural network architecture called *skipping recurrent neural networks* (S-RNNs) (Sigurdsson, Chen, and Gupta 2017). Whereas conventional recurrent neural networks such as LSTMs and sequence-to-sequence have limited ability to learn a distribution conditioned on a history of observations, S-RNNs learn to select a set of $k$ elements in a sequence that minimizes the amount of information lost by removing all other items. When the model is applied to a new sequence, the set of $k$ items selected by the model is the best summary of the overall sequence.

The S-RNN neural network architecture was originally developed to summarize image albums. The goal of the S-RNN network is to learn the maximum likelihood model parameters ($\mathcal{M}$) by maximizing the marginal likelihood of the observed data according to the following objective function:

$$\mathcal{M}^* = \arg\max_{\mathcal{M}} \log \sum_{z_{1:N}} P(x_{1:T}, z_{1:N}|\mathcal{M}) - \lambda\mathcal{R}(\mathcal{M}) \quad (2)$$

where $x_{1:T}$ is the $T$ images in an album, $z_{1:N}$ is the set of indexes that represent the images selected for the summary, $N$ is the number of elements in the summary ($N \ll T$), and $\mathcal{R}(\cdot)$ is the $\ell_2$ regularizer. Maximizing the marginal likelihood over all possible subsets of $z$ is done with Expectation Maximization (EM). See (Sigurdsson, Chen, and Gupta 2017) for details for how the expectation and maximization steps are performed. Each index $z_i$ generated by the network is the index of an image within an image album. A summary is produced by retrieving the image at each index, retaining the order, and discarding the rest.

We modified S-RNN to work with textual data. We observe that, from the perspective of the S-RNN algorithm, a story in event representation can be considered analogous to an image album, an event is analogous to a single image, and each of the four words in an event is analogous to a pixel but with a large-but-finite set of possible values in the form of a one-hot vector word encoding instead of RGB encoding. Thus, our S-RNN model is trained with a corpus of story events, identical to that used by MCMC and the sequence-to-sequence network. Once the S-RNN model is trained, arbitrary sequences of events can be input and $k$ indices are generated. A story summary is generated by retrieving each of the $k$ events at the specified position in the story.

To use S-RNN generated summaries as acceptance criteria, we first reduce the summary to a list of $k$ verbs—we simply discard the subjects, objects, and modifiers from each event. Although it would be possible to perform an exact event match, we felt it would be too restrictive and allowing MCMC to only match verbs would give it some flexibility. Summaries can be generated for every story in the same story event corpus that MCMC and the sequence-to-sequence network uses. Any number of summaries can be generated in this way; however in our case study we use two of the most frequent summaries. For each summary, a different MCMC distribution is trained. At the beginning of the story generation process, we randomly pick one of the distributions and then randomly sample from it until a story of the desired length has been generated.

# Case Study

In this section we present a case study in which we use our technique in conjunction with romance movie summaries taken from Wikipedia to generate new stories that adhere to desired criteria. The goal of this study is to determine how

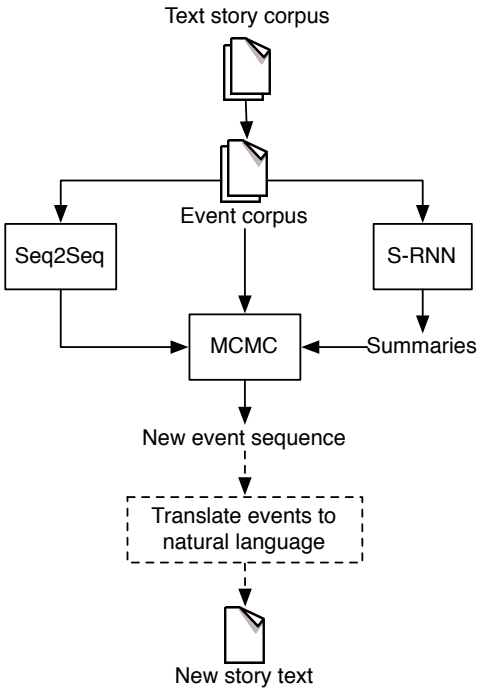


Figure 2: The story generation pipeline.

well this technique can produce stories that adhere to the acceptance criteria used during the MCMC sampling.

Our system architecture is shown in Figure 2. The dashed lines and boxes represent future work. Since events are not easily human readable, we must translate events generated by MCMC back into human-readable form. For example, Martin et al. (2017) use a neural network to translate events into natural language.

## Event Corpus

As seen in Figure 2, we first convert each sentence in the story corpus into the event representation discussed previously. This can be accomplished in linear time using the Stanford CoreNLP toolkit (Manning et al. 2014), Wordnet (Miller 1995), and Verbnet (Schuler 2005) as described in (Martin et al. 2017). During conversion, named entities are removed and replaced with general identifiers, remaining nouns are converted into high-level classes defined using Wordnet synsets, and verbs were assigned to verb classes using Verbnet. This is done to reduce event sparsity and make learning the distribution easier. An example sentence and its corresponding event representation is shown in Figure 3. This event corpus is then used to train both the sequence-to-sequence event succession model and the S-RNN, and it is used to generate the initial bigram distribution used to produce candidate stories during sampling.

## Acceptance Criteria

The sequence-to-sequence transition model and the S-RNN are used to calculate the acceptance score used during sampling. The S-RNN is used to generate a sequence of three

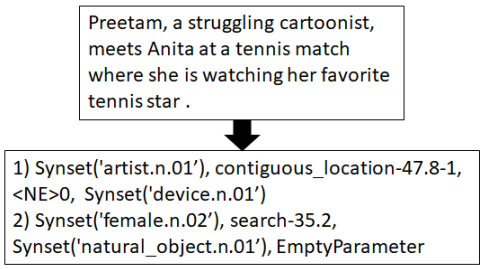


Figure 3: Example sentence and its conversion into two separate events. <NE> refers to a named entity, Synsets were assigned using Wordnet, and verb classes were assigned using Verbnet.

events for each story in the training set. These sequences act as summaries of the actions that take place in story seen during training. We use these summaries to impose strict restrictions the type of stories that can be accepted during sampling. To simplify this criteria, we are only concerned with the verb element of the event rather than the entire event. This makes it easier for the MCMC sampling algorithm to generate stories that adhere to this criterion. In this case study, we use two sequences in particular:

1. amuse-31.1, get-13.5.1-1, fit-54.3
2. transfer-mesg-37.1.1-1-1, seem-109-1-1, become-109.1

Since these sequences were generated from our event corpus, they consist of verb classes from VerbNet rather than the natural language text from the original plot corpus. Only stories in which these three verb classes occur in this correct order relative to each other will be accepted.

The LSTM neural network is used to ensure that the sampled stories retain a reasonable structure. Without this additional criteria, the MCMC sampler would eventually converge upon a story that simply recreates the verb ordering criterion discussed previously. For our experiments, we trained a LSTM neural network with two layers and embedding layer of size 300 for 100 epochs using the same training data that was used to train the S-RNN. Each sampled story that adheres to the verb ordering criteria is then run through this network. We then extract the log-probability that the sampled story could have been produced by the network and use this to score the story. If this story's score is greater the previous accepted story's score, then the current story is accepted. Otherwise, the story is accepted according to the probability of acceptance, $P(a)$: $P(a) = \frac{P(x|X^t)}{P(y|X^{t-1})}$. Here $P(x|X^t)$ is the log probability of sampling the current story from the current distribution, and $P(y|X^{t-1})$ is the log probability of the most recently accepted story.

## MCMC Sampling

The event corpus is used to populate the initial probability distribution used for sampling. To do this, we scan each story in the corpus and calculate the conditional probability of each word given the previous word in the story using word frequency. During training, the sampling algorithm creates

Table 1: Percentage of stories that adhere to the acceptance criteria before MCMC sampling and after.

| Condition | Criteria 1 | Criteria 2 |
|---|---|---|
| Original Distribution | 1.0% | 1.2% |
| After MCMC Sampling | 85.5% | 86.1% |

stories by sampling this distribution until 100,000 stories have been accepted. For our experiments, we limited our algorithm to generate only stories containing 10 events. This was done to prevent the sampling algorithm from creating overlong, possibly rambling, stories as well as to examine if the algorithm would be able to successfully guide the story using a small, fixed number of events. Each time a story is accepted, the sampling distribution is updated with this new story's information and this new distribution is used to sample the next stories.

For this study, we created two models with different acceptance criteria and then used them to generate one thousand stories for testing. To evaluate our approach we calculated the percentage of these stories that adhered to the verb ordering criteria. For comparison, we also calculated the percentage of stories in the event corpus that adhered to the original ordering criteria.

### Results and Discussion

The results of this evaluation can be seen in Table 1. These results show that stories sampled from the conditional word distribution created using our MCMC sampling technique can accurately recreate the verb ordering criteria around 85% of the time. We also show how many stories in the original learning corpus adhere to each of the verb ordering criteria used for testing. In the original corpus, only 1% of stories contained the provided verbs in the desired order. This shows that our technique was able to significantly shift the starting distribution towards the acceptance criteria.

The primary conclusion to be drawn from these results is that the MCMC sampling algorithm that we introduced appears to successfully produce a word distribution that produces stories that fit the provided acceptance criteria. For each criteria we examined, our technique was able to sample stories that fit the criteria 85.5% of the time and 86.1% for each criterion, respectively. This is especially notable given how few of the stories in the original corpus adhered to each of these criteria. This means that this technique was able to detect the signal and effectively shift the distribution to improve the probability that these stories would be sampled.

While this shift is promising, these results still show that the random nature of sampling stories from this distribution can result in some stories not adhering to the desired criteria. While the goal of this technique is to produce a distribution that greatly increases the probability of sampling an acceptable story, it is still possible to generate stories that do not adhere to the acceptance criteria. This could possibly be improved by encoding a longer event history into the sampling distribution. So, instead of encoding bigrams into the distribution we could encode trigrams or four-grams. This would increase sparsity in the distribution which could increase the probability that the resultant stories exhibit the desired properties. This would, however, likely result in fewer unique stories that could be generated. This tradeoff is important to consider when deciding on how much story history to encode in the sampling distribution. This could also be helped by improving the quality of the LSTM used as part of our acceptance criteria. Extending its capabilities beyond bigram prediction should also, albeit indirectly, improve the quality of stories generated using our technique.

## Limitations and Future Work

Despite the effectiveness of this technique, there are some important limitations that need to be considered. The most important of these is that the resultant stories are not, in their current form, semantically interpretable. This makes it hard to easily determine the quality of the sentences sampled using our approach beyond verifying that they exhibit the author-defined story properties. This is because we use an event representation to simplify the sampling process. As shown in Figure 2, the final module in our system pipeline translates sequences of events back into natural language. This, however, is a nontrivial task as this module would need to possess the ability to ground the abstract noun and verb classes we define in natural language in a way that is human readable and still maintains story context.

In addition, it is not clear how to define effective acceptance criteria, or how to define more complex acceptance criteria. In our case study, we examine one of the simpler cases where we are only concerned with generating stories that have high-level plot structure similar to existing stories within a genre, as defined by the event summaries learned by S-RNN. We have not explored how to construct more complex acceptance criteria that could give authors more control over how a story should unfold.

## Conclusions

In this work, we propose a MCMC sampling technique for guided, open story generation using naturally occurring story corpora. We show how MCMC sampling, when combined with a set of criteria for accepting or rejecting sampled stories, can create a probability distribution over words that can be used to generate stories that adhere to the acceptance criteria. We also perform a case study using this technique to generate stories based on movie plots taken from Wikipedia. This case study shows that our technique was successful at generating stories that adhered to our provided acceptance criteria about 85% of the time.

While this work is still in a preliminary stage, we feel that it shows great potential for story generation. This is because it is able to easily take advantage of large, naturally occurring story corpora while also being able to provide authors a way to direct the story as they see fit.

## Acknowledgements

This work was supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. W911NF-15-C-0246 and by the National Science Foundation under Grant No. IIS-1350339.

## References


Cavazza, M.; Charles, F.; and Mead, S. 2002. Planning characters' behaviour in interactive storytelling. *Journal of Visualization and Computer Animation* 13:121–131.

Chib, S., and Greenberg, E. 1995. Understanding the metropolis-hastings algorithm. *The american statistician* 49(4):327–335.

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

Farrell, R., and Ware, S. 2016. Fast and diverse narrative planning through novelty pruning. In *Proceedings of the 12th AAAI International Conference on Artificial Intelligence 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(4–5):235–242.

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

Khalifa, A.; Barros, G. A.; and Togelius, J. 2017. Deeptingle. *arXiv preprint arXiv:1705.03557*.

Lebowitz, M. 1987. Planning stories. In *Proceedings of the 9th Annual Conference of the Cognitive Science Society*, 234–242.

Li, B., and Riedl, M. O. 2010. An offline planning approach to game plotline adaptation. In *Proceedings of the 6th Conference on Artificial Intelligence for Interactive Digital Entertainment Conference*, 45–50.

Li, B.; Lee-Urban, S.; Johnston, G.; and Riedl, M. O. 2013. Story generation with crowdsourced plot graphs. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence*.

Manning, C. D.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S. J.; and McClosky, D. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, 55–60.

Martens, C.; Ferreira, J.; Bosser, A.-G.; and Cavazza, M. 2014. Generative story worlds as linear logic programs. In *Proceedings of the 2014 AAAI Workshop on Intelligent Narrative Technologies*.

Martin, L. J.; Ammanabrolu, P.; Hancock, W.; Singh, S.; Harrison, B.; and Riedl, M. O. 2017. Event representations for automated story generation with deep neural nets. In *arXiv:1706.01331*.

Meehan, J. R. 1977. TALE-SPIN: An interactive program that writes stories. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, 91–98.

Miller, G. A. 1995. WordNet: A lexical database for english. *Communications of the ACM* 38(11):39–41.

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* 13:119–139.

Porteous, J., and Cavazza, M. 2009. Controlling narrative generation with planning trajectories: the role of constraints. In *Proceedings of the 2nd International Conference on Interactive Digital Storytelling*, 234–245.

Riedl, M. O., and Young, R. M. 2010. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research* 39:217–268.

Roemmele, M. 2016. Writing stories with help from recurrent neural networks. In *AAAI*, 4311–4342.

Schuler, K. K. 2005. Verbnet: A broad-coverage, comprehensive verb lexicon.

Sharples, M. 1999. *How We Write: Writing as Creative Design*. London: Routledge.

Sigurdsson, G. A.; Chen, X.; and Gupta, A. 2017. Learning visual storylines with skipping recurrent neural networks.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 3104–3112.

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. R. 1994. *The Creative Process: A Computer Model of Storytelling*. Hillsdale, NJ: Lawrence Erlbaum Associates.