Event Representations for Automated Story Generation with Deep Neural Nets

Lara J. Martin, Prithviraj Ammanabrolu, William Hancock,
Shruti Singh, Brent Harrison, and Mark O. Riedl
School of Interactive Computing, Georgia Institute of Technology
Atlanta, Georgia
[ljmartin;raj.ammanabrolu;whancock;shruti.singh;brent.harrison;riedl]@gatech.edu

ABSTRACT

Automated story generation is the problem of automatically selecting a sequence of events, actions, or words that can be told as a story. We seek to develop a system that can generate stories by learning everything it needs to know from textual story corpora. To date, recurrent neural networks that learn language models at character, word, or sentence levels have had little success generating coherent stories. We explore the question of event representations that provide a mid-level of abstraction between words and sentences in order to retain the semantic information of the original data while minimizing event sparsity. We present a technique for preprocessing textual story data into event sequences. We then present a technique for automated story generation whereby we decompose the problem into the generation of successive events (*event2event*) and the generation of natural language sentences from events (*event2sentence*). We give empirical results comparing different event representations and their effects on event successor generation and the translation of events to natural language.

CCS CONCEPTS

•**Artificial Intelligence** → *Machine Learning; Natural Language Processing; Neural Networks;*

KEYWORDS

story generation, event extraction, deep neural networks, artificial intelligence

ACM Reference format:
Lara J. Martin, Prithviraj Ammanabrolu, William Hancock,
Shruti Singh, Brent Harrison, and Mark O. Riedl. 2017. Event Representations for Automated Story Generation with Deep Neural Nets. In *Proceedings of SIGKDD Workshop on Machine Learning for Creativity, Halifax, Nova Scotia, Canada, August 2017 (SIGKDD 2017),* 8 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Automated story generation is the problem of automatically selecting a sequence of events, actions, or words that can be told as a story. To date, most story generation systems have used symbolic planning [10, 14, 20, 23, 25] or case-based reasoning [6]. While these automated story generation systems were able to produce impressive results, they rely on a human-knowledge engineer to provide symbolic domain models that indicated legal characters, actions, and knowledge about when character actions can and cannot be performed; these systems are limited to only telling stories about topics that are covered by the domain knowledge. Consequently, it is difficult to determine whether the quality of the stories produced by these systems is a result of the algorithm or good knowledge engineering.

Open story generation [11] is the problem of automatically generating a story about any topic without *a priori* manual domain knowledge engineering. Open story generation requires an intelligent system to either learn a domain model from available data [11, 26] or to reuse data and knowledge available from a corpus [30].

In this paper, we explore the use of recurrent encoder-decoder neural networks (e.g., *Sequence2Sequence* [29]) for open story generation. A recurrent encoder-decoder neural network is trained to predict the next token(s) in a sequence, given one or more input tokens. The network architecture and set of weights θ comprise a generative model capturing and generalizing over patters observed in the training data. For open story generation, we must train the network on a dataset that encompasses as many story topics as possible. For this work, we use a corpus of movie plot summaries extracted from Wikipedia [2] under the premise that the set of movies plots on Wikipedia covers the range of topics that people want to tell stories about.

In narratological terms, an *event* is a unit of story featuring a world state change [24]. Textual story corpora, including Wikipedia movie plot corpora, is comprised of unstructured textual sentences. One benefit to dealing with movie plots is its clarity of events that occur, although this is often to the expense of more creative language. Even so, character- or word-level analysis of these sentences would fail to capture the interplay between the words that make up the meaning behind the sentence. Character- and word-level recurrent neural networks can learn to create grammatically correct sentences but often fail to produce coherent narratives beyond a couple of sentences. On the other hand, sentence-level events would be too unique from each other to find any real relationship between them. Even with a large corpus of stories, we would most likely have sequences of sentences that would only ever be seen once. For example, "Old ranch-hand Frank Osorio travels from Patagonia to Buenos Aires to bring the news of his daughter's demise to his granddaughter Alina." occurs only once in our corpus, so we have only ever seen one example of what is likely to occur

DOI: 10.1145/nnnnnnn.nnnnnnn

before and after it (if anything). Due to event sparsity, we are likely to have poor predictive ability.

In order to help maintain a coherent story, one can provide an event representation that is expressive enough to preserve the semantic meaning of sentences in a story corpus while also reducing the sparsity of events (i.e. increasing the potential overlap of events across stories and the number of examples of events the learner observes). In this paper, we have developed an event representation that aids in the process of automated, open story generation. The insight is that if one can extract some basic semantic information from the sentences of preexisting stories, one can learn the skeletons of what "good" stories are supposed to be like. Then, using these templates, the system will be able to generate novel sequences of events that would resemble a decent story.

The first contribution of our paper is thus an event representation and a proposed recurrent encoder-decoder neural network for story generation called *event2event*. We evaluate our event representation against the naive baseline sentence representation and a number of alternative representations.

In *event2event*, a textual story corpus is preprocessed—sentences are translated into our event representation by extracting the core semantic information from each sentence. Event preprocessing is a linear-time algorithm using a number of natural language processing techniques. The processed text is then used to train the neural network. However, event preprocessing is a lossy process and the resultant events are not human-readable. To address this, we present a story generation pipeline in which a second neural network, *event2sentence*, translates abstract events back into natural language sentences. The *event2sentence* network is an encoder-decoder network trained to fill in the missing details necessary for the abstract events to be human-readable.

Our second contribution is the overall story generation pipeline in which subsequent events of a story are generated via an *event2event* network and then translated into natural language using an *event2sentence* network. We present an evaluation of *event2sentence* on different event representations and draw conclusions about the effect of representations on the ability to produce readable stories.

The remainder of the paper is organized as follows. In Section 2 we discuss related work on automated story generation. In Section 3 we introduce our event representation. Section 4 introduces our *event2event* network and provides an evaluation of the event representation in the context of story generation. Section 5 shows how the event representation can be used in *event2sentence* to generate a human-readable sentences from events. We end with a discussion of future work and conclusions about these experiments and how our event representation and event-to-sentence model will fit into our final system.

2 RELATED WORK

Automated Story Generation has been a research problem of interest since nearly the inception of artificial intelligence. Early attempts relied on symbolic planning [10, 14, 20, 25] or case-based reasoning using ontologies [6]. These techniques could only generate stories for predetermined and well-defined domains of characters, places, and actions. The creativity of these systems conflated the robustness of manually-engineered knowledge and algorithm suitability.

Recently, machine learning has been used to attempt to learn the domain model from which stories can be created or to identify segments of story content available in an existing repository to assemble stories. The *SayAnthing* system [30] uses textual case-based reasoning to identify relevant existing story content in online blogs. The *Scheherazade* system [11] uses a crowdsourced corpus of example stories to learn a domain model from which to generate novel stories.

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 output tokens, stories can be generated by sampling from the output of a recurrent neural network trained on story text data. Roemmele and Gordon [26] use a Long Short-Term Memory (LSTM) network [7] to generate stories. They use Skip-thought vectors [9] to encode sentences and a technique similar to word2vec [15] to embedded entire sentences into 4,800-dimensional space. They trained their network on the BookCorpus dataset. Khalifa et al. [8] 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*.

Based off of the theory of script learning [27], Chambers and Jurafsky [3] learn causal chains that revolve around a protagonist. They developed a representation that took note of the event/verb that occurred and the type of dependency that connected the event to the protagonist (e.g. was the protagonist the object of this event?).

Pichotta and Mooney [21] developed a 5-tuple event representation of (v, e_s, e_o, e_p, p), where v is the verb, p is a preposition, and the e's are nouns representing the subject, direction object, and prepositional object, respectively. Our representation was inspired by this work, although we use a slightly different representation. Because it was a paper on script learning, they did not need to convert the event representations back into natural language.

Related to automated story generation, the *story cloze test* [17] is the task of choosing between two given endings to a story. The story cloze test transforms story generation into a classification problem: a 4-sentence story is given along with two alternative sentences that can be the 5th sentence. State-of-the art story cloze test techniques use a combination of word embeddings, sentiment analysis, and stylistic features [18].

3 EVENT REPRESENTATION

Automated story generation can be formalized as follows: given a sequence of events, sample from the probability distribution over successor events. That is, simple automated story generation can be expressed as a process whereby the next event is computed by sampling or maximizing $Pr_\theta(e_{t+1}|e_{t-k}, ..., e_{t-1}, e_t)$ where θ is the set of parameters of a generative domain model, e_i is the event at time i, and where k indicates the size of a sliding window of context, or history.

In our work, the probability distribution is produced by a recurrent encoder-decoder network with parameters θ. In this section, we consider what the level of abstraction for the inputs into the network should be such that it produces the best predictive power while retaining semantic knowledge. Event sparsity results in a situation where all event successors have a low probability of occurrence, potentially within a margin of error. In this situation, story generation devolves to a random generation process.

Following Pichotta and Mooney [21], we developed a 4-tuple event representation $\langle s, v, o, m \rangle$ where v is a verb, s is the subject of the verb, o is the object of the verb, and m is the modifier or "wildcard", which can be a propositional object, indirect object, causal complement (e.g., In "I was glad that he drove," "drove" is the causal complement to "glad."), or any other dependency unclassifiable to Stanford's dependency parser. All words were stemmed.

A textual story corpus is preprocessed to convert sentences into events in linear time. Events are created by first extracting dependencies with Stanford's CoreNLP [12] and locating the appropriate dependencies mentioned above. If the object or modifier cannot be identified, we insert the placeholder $EmptyParameter$.

Our event translation process can either extract a single event from a sentence or multiple events per sentence. If we were to extract multiple events, it is because there are conjunctions in the sentence. Consider the sentence "John and Mary went to the store," our algorithm would extract two events: $\langle john, go, store, EmptyParameter \rangle$ and $\langle mary, go, store, EmptyParameter \rangle$. As we process a sentence's dependency tree, we maintain a set of stacks such that if the two parts of the sentence were connected by the verb (such as in the example) the last verb was popped from the stack. The same was done for sentential clauses. Throughout the experiments reported in this paper, the average number of events per sentence was 2.69.

Our experiments below used a corpus of movie plots from Wikipedia [2], which we cleaned to any remove extraneous Wikipedia syntax, such as links for which actors played which characters. This corpus contains 42,170 stories with the average number of sentences per story being 14.515. In this corpus, there is minimal character dialogue. How to treat dialogue in story generation is an open question. We suggest that dialogue can be removed and replaced with a single "dialogue" event. However, identifying who is speaking in story text is a hard natural language processing problem that is not fully solved (c.f., [4]). For the time being, we parsed dialogue through the same process as the rest of the data.

The simplest form of our event representation is achieved by extracting the verb, subject, object, and modifier term from each sentence. However, there are variations on the event representation that increase the level of abstraction (and thus decrease sparsity) and help the encoder-decoder network predict successor events. We enumerate some of the possible variations below.

- **Generalized.** Each element in the event tuple undergoes further abstraction. Named entities are identified (cf. [5]) and replaced with the tag <NE>n, where n indicates the n-th named entity in the sentence. The rest of the nouns were replaced by the WordNet [16] synset two levels up in the inherited hypernym hierarchy, giving us a general category (e.g. self-propelled_vehicle.n.01 vs the original word "car" (car.n.01)), while avoiding labeling it too generally (e.g. entity.n.01). Verbs were replaced by VerbNet [28] version 3.2.4 [1] frames (e.g. "arrived" becomes "escape-51.1", "transferring" becomes "contribute-13.2-2").
- **Named Entity Numbering.** There were two ways of numbering the named entities (i.e. people's names) that we experimented with. One way had the named entity numbering reset with every sentence (consistent within sentence)–or, *sentence NEs*, our "default". The other way had the numbering reset after every input-output pair (i.e. every line of data)–or, *continued NEs*.
- **Adding Genre Information.** We did topic modeling on the entire corpus using Python's Latent Dirichlet Analysis (LDA) algorithm [2] set for discovering 100 different categories. We took this categorization as a type of emergent genre classification. Some clusters had a clear pattern, e.g., "job company work money business". Others, were less clear. Each cluster was given a unique number and this genre number was added to the event representation to create a 5-tuple $\langle s, v, o, m, g \rangle$ where s, v, o, and m are defined as above and g is the genre cluster number.

We note that other event representations can exist, including representations that incorporate more information as in [21]. The experiments in the next section show how different representations affect the ability of a recurrent neural network to predict story continuations.

4 EVENT-TO-EVENT

The *event2event* network is a recurrent multi-layer encoder-decoder network based on [29]. Unless otherwise stated in experiments below, our *event2event* network is trained with input $x = w_1^n, w_2^n, w_3^n, w_4^n$ where each w_i^n is either s, v, o, or m from event n, and output $y = w_1^{n+1}, w_2^{n+1}, w_3^{n+1}, w_4^{n+1}$.

The experiments described below seek to determine how different event representations affected *event2event* predictions of the successor event in a story. We evaluated each event representation using two metrics. *Perplexity* is the measure of how "surprised" a model is by a training set. Perplexity is a metric which is normally used to evaluate language models. Here we use it to gain a sense of how well the probabilistic model we have trained can predict the data. Specifically, we built the model using an n-gram length of 1:

$$Perplexity = 2^{-\sum_x p(x) \log_2 p(x)} \tag{1}$$

where x is a token in the text, and

$$p(x) = \frac{count(x)}{\sum_x count(x)} \tag{2}$$

The larger the unigram perplexity, the less likely a model is to produce the next unigram in a test dataset.

The second metric is BLEU score, which compares the similarity between the generated output and the "ground truth" by looking at n-gram precision. The neural network architecture we use was initially envisioned for machine translation purposes, where BLEU is a common evaluation metric. Specifically, we use an n-gram length of 4 and so the score takes into account all n-gram overlaps

[1] https://verbs.colorado.edu/vn3.2.4-test-uvi/index.php
[2] https://pypi.python.org/pypi/lda

between the generated and expected output where n varies from 1 to 4 [19].

We use a greedy decoder to produce the most likely sequence given an input sequence.

$$\hat{E} = \arg\max_{e} Pr(e|S) \tag{3}$$

where $\hat{E}$ is the generated sequence, S is the input sequence, and e represents the possible output sequences.

4.1 Experimental Setup

For each experiment, we trained a long short-term memory (LSTM) sequence-to-sequence recurrent neural net (RNN) [29] using Tensorflow [1]. Each RNN was trained with the same parameters (0.5 learning rate, 0.99 learning rate decay, 5.0 maximum gradient, 64 batch size, 1024 model layer size, and 4 layers), varying only the input/output, the bucket size, the number of epochs and the vocabulary. The neural nets were trained until the decrease in overall loss was less than 5% per epoch. This took between 40 to 60 epochs for all experiments.

The data was split into 80% training, 10% validation, and 10% test data. All reported results were evaluated using the the held-out test data.

We evaluated 11 versions of our event representation against a sentence-level baseline. Numbers below correspond to rows in results Table 1.

(0) *Original Sentences.* As our baseline, we evaluated how well an original sentence can predict its following original sentence within a story.
(1) *Original Words Baseline.* We took the most basic event representation. This is the 4-word event representation introduced in Section 3: $\langle subject, verb, direct\_object, modifier\rangle$ with no abstraction and using original named entity names.
(2) *Original Words with <NE>s.* Identical to the previous experiment except entity names that were classified as "PERSON" through NER were substituted with <NE>*n*. Other named entities were given their respective NER tags (e.g. named entities classified as LOCATION were substituted with "LOCATION").
(3) *Generalized.* The same 4-word event structure except with named entities replaced and all other words generalized through WordNet or VerbNet, following the procedure from Section 4.

To avoid an overwhelming number of experiments, the next set of experiments used the "winner" of the first set of experiments. Subsequent experiments used variations of the generalized event representation (#3), which showed drastically lower perplexity scores.

(4) *Generalized, Continued <NE>s.* This experiment mirrors the previous with the exception of the number of the <NE>s. In the previous experiment, the numbers restarted after every event. Here, the numbers continue across input and output. So if $event_1$ mentioned "Kendall" and $event_2$ (which follows $event_1$ in the story) mentioned "Kendall", then both would have the same number for this character.
(5) *Generalized + Genre.* This is the same event structure as experiment #3 with the exception of an additional, 5^{th} parameter in the event: genre. Genre numbers are explained in Section 3. The genre number was used in training for *event2event* but removed from inputs and outputs before testing; it artificially inflated BLEU scores because it was easy for the network to guess the genre number as the genre number was weighted equally to other words.
(6) *Generalized Bigram.* This experiment tests whether RNN history aids in predicting the next event. We modified *event2event* to give it the event bigram e_{n-1}, e_n and to predict e_{n+1}, e_{n+2}. We believe that this experiment could generalize to cases with a $e_{n-k}, ..., e_n$ history.
(7) *Generalized Bigram, Continued <NE>s.* This experiment has the same continued NE numbering as experiment #4 had but we trained *event2event* with event bigrams.

The following four experiments investigate extracting more than one event per sentence in the story corpus when possible; the prior experiments only use the first event per sentence in the original corpus.

(8) *Generalized Multiple, Sequential.* When a sentence yields more than one event, $e_n^1, e_n^2, ...$ where n is the nth sentence and e_n^i is the ith event created from the nth sentence, we train the neural network as if each event occurs in sequence, i.e., e_n^1 predicts e_n^2, e_n^2 predicts e_n^3, etc. The last event from sentence n predicts the first event from sentence $n+1$.
(9) *Generalized Multiple, Any Order.* Here we gave the RNN all orderings of the events produced by a single sentence paired, in turn, with all orderings of each event of the following sentence.
(10) *Generalized Multiple, All to All.* In this experiment, we took all of the events produced by a single sentence together as the input, with all of the events produced by its following sentence together as output. For example, if sentence i produced events e_i^1, e_i^2, and e_i^3, and the following sentence j produced events e_j^1 and e_j^2, then we would train our neural network on the input: $e_i^1\ e_i^2\ e_i^3$, and the output: $e_j^1\ e_j^2$.
(11) *Generalized Bigram + Genre.* This is a combination of the ideas from experiments #5 and #6: generalized events in event bigrams and with genre added.

4.2 Results and Discussion

The results from the experiments outlined above can be found in Table 1.

The original word events had similar perplexity to original sentences. This parallels similar observations made by Pichotta and Mooney [22]. Deleting words did little to improve the predictive ability of our *event2event* network. However, perplexity improved significantly once character names were replaced by generalized <NE>tags, followed by generalizing other words and verbs.

Overall, the generalized events had much better perplexity scores, and making them into bigrams—incorporating history—improved the BLEU scores to nearly those of the original word events. Adding in genre information improved perplexity.

The best perplexity was achieved when multiple generalized events were created from sentences as long as all of the events were fed in at the same time (i.e. no order was being forced upon the events that came from the same sentence). The training data was set up to encourage the neural network to correlate all of the events in one sentence with all of the events from the next sentence.

Although the "specific" events and original sentences performed better in terms of BLEU score, it is our belief that BLEU is not the most appropriate metric for story generation because it emphasizes the recreation of the input. Overall, BLEU scores are very low for all experiments, attesting to the inappropriateness of the metric. Perplexity is a more appropriate metric for story generation because it correlates with the ability for a model to predict the entire test dataset. Borrowing heavily from the field of language modeling, the recurrent neural network approach to story generation is a prediction problem.

Our intuition that the generalized events would perform better in generating successive events bears out in the data. However, greater generalization makes it harder to return events to natural language sentences. Section 5 addresses this question. We also see that the BLEU scores for the bigram experiments are generally higher than the others. This shows that history matters and that the additional context provided increases the number of n-gram overlaps between the generated and expected outputs.

The movie plots corpus contains numerous sentences that can be interpreted as describing multiple events. Naive implementation of multiple events hurt perplexity because there is no implicit order of events generated from the same sentence; they are not necessarily sequential. When we allow multiple events from sentences to be followed by all of the events from a subsequent sentence, perplexity improves. We show examples of events generated by *event2event* with the all-to-all event representation in Table 2. It can be hard to manually interpret events. One can take any event in the table and create a sentence from it by making each abstract category more specific. For example, the event ⟨substance.n.01, enforce-63, whole.n.02, EmptyParameter⟩ could, hypothetically, have come from the sentence (or sentence fragment) "the water stopped everything." In actuality, this is the final event from the sentence "at the hospital ginny finally stirs and dave deduces that steve s silicon solution can be used to control the rocks," specifically the second clause, "steve s silicon solution can be used to control the rocks."

Table 1: Results from the event-to-event experiments.

Experiment	**Perplexity**	**BLEU**
(0) Original Sentences	704.815	0.0432
(1) Original Words Baseline	748.914	**0.1880**
(2) Original Words with <NE>s	166.646	0.1878
(3) Generalized Baseline	**54.231**	0.0575
(4) Generalized, Continued NEs	56.180	0.0544
(5) Generalized + Genre	**48.041**	0.0525
(6) Generalized Bigram	50.636	0.1549
(7) Generalized Bigram, Continued NEs	50.189	**0.1567**
(8) Generalized Multiple, Sequential	58.562	0.0521
(9) Generalized Multiple, Any Order	61.532	0.0405
(10) Generalized Multiple, All to All	**45.223**	0.1091
(11) Generalized Bigram + Genre	48.505	**0.1102**

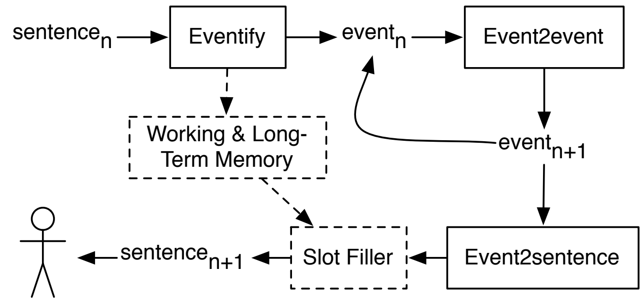

Figure 1: Our automated story generation pipeline. Dashed boxes and arrows represent future work.

5 EVENT-TO-SENTENCE

The *event2event* network takes an input event $e_n = \langle s^n, v^n, o^n, v^n \rangle$ and samples from a distribution over possible successor events $e_{n+1} = \langle s^{n+1}, v^{n+1}, o^{n+1}, m^{n+1} \rangle$. Unfortunately, events are not human-readable and must be converted to natural language sentences. While the conversion from sentences to (multiple) events for *event2event* is a linear and lossy process, the translation of events back to sentences is non-trivial as it requires adding details back in. Complicating the situation, the *event2event* encoder-decoder network is not guaranteed to produce an event that has ever been seen in the training story corpus. Further complicating the situation, experiments with event representations for *event2event* indicate that greater generalization lends to better story generation. Unfortunately, the greater the generalization, the harder it is to translate an event back into a natural language sentence.

In this section we introduce *event2sentence*, a neural network designed to translate an event into natural language. As before, we use a recurrent encoder-decoder network based on [29]. The *event2sentence* network is trained on parallel corpora of sentences from a story corpus and the corresponding events. In that sense, *event2sentence* is attempting to learn to reverse the lossy event creation process.

We envision *event2event* and *event2sentence* working together as illustrated in Figure 1. First, a sentence—provided by a human—is turned into one or more events. The *event2event* network generates one or more successive events. The *event2sentence* network translates the events back into natural language and presents it to the human reader. The dashed lines and boxes represent future work (see Section 6). To continue story generation, $event_{n+1}$ can be fed back into *event2event*; the sentence generation is purely for human consumption.

The *event2sentence* experiments in the next section investigate how well different event representations can be "translated" back into an natural language sentences.

5.1 Experimental Setup

The LSTM RNNs were setup with the same parameters as the event-to-event experiments (0.5 learning rate, 0.99 learning rate decay, 5.0 maximum gradient, 64 batch size, 1024 model layer size, and 4 layers), so that we could compare our results to the original sentence experiment (see Section 4.1). Other parameters such as:

Table 2: Sample *event2event* output from the All to All generalized event experiment (#10), comparing expected and generated output events. In this example, we feed in 4 events (column 1) taken from a single sentence in a story. The next sentence in the original story has the events found in column 2. The output of our system is in column 3.

Input (All Events from One Sentence)	Expected Output (Events from the Following Sentence)	Generated Output
⟨physical_entity.*n*.01, say-37.77-1, act.*n*.02, EmptyParameter⟩ ⟨act.*n*.02, wish-62, evacuate, EmptyParameter⟩ ⟨act.*n*.02, banish-10.2, administrative_district.*n*.01, EmptyParameter⟩ ⟨weather.*n*.01, lodge-46, EmptyParameter, EmptyParameter⟩	⟨<NE>0, discover-84, EmptyParameter, EmptyParameter⟩ ⟨substance.*n*.01, fit-54.3, control, EmptyParameter⟩ ⟨substance.n.01, enforce-63, whole.n.02, EmptyParameter⟩	⟨<NE>0, transfer_mesg-37.1.1-1-1, <NE>1, EmptyParameter⟩ ⟨seem-109-1-1, EmptyParameter, EmptyParameter, abstraction.*n*.06⟩ ⟨abstraction.*n*.06, manner_speaking-37.3, EmptyParameter, EmptyParameter⟩

evaluation metrics, the training, validation, testing split in the data, the method of deciding the number of epochs, and the method of decoding were also the same.

As a baseline, we trained the *event2sentence* network to recreate the input sentence. For the remainder of experiments, the input was events of a particular representation and output was a newly-generated sentence based on the input event. The models in these experiments were trained on the events paired with the sentences they were "eventified" from. In a complete story generation system, the output of the *event2event* network feeds into the *event2sentence* network. However, for this paper, we tested the *event2sentence* network on the same events extracted from the original sentences as were used for *event2event* in order to conduct controlled experiments and compute perplexity and BLEU scores.

To test *event2sentence* with an event representations that used the original words is relatively straight forward. Experimenting on translating generalized events to natural language sentences was more challenging. Initial experiments produced very poor results because the network would have to guess what character name to substitute back in for <NE>tags. Likewise, the network would have to guess which of the many verbs to use from a VerbNet frame and which of many nouns to use from a WordNet synset hierarchy.

We devised an alternative approach to *event2sentence* whereby sentences were first partially eventified. That is, we trained *event2sentence* on generalized sentences where the "PERSON" named entities were replaced by <NE>tags, other named entities were replaced by their NER category, and the remaining nouns were replaced with WordNet synsets. The verbs were left alone since they often do not have to be consistent across sentences within a story. The intuition here is that the character names and particulars of objects and places are highly mutable and do not affect the overall flow of a story as long as they remain consistent. The use of partially generalized sentences can be an advantage to storytelling because names and other details can be added and made consistent in post-processing.

Below, we show an example of a sentence and its partially generalized counterpart. The original sentence

> The remaining craft launches a Buzz droid at the ARC 1 7 0 which lands near the Clone Trooper rear gunner who uses a can of Buzz Spray to dislodge the robot.

would be partially generalized to

> The remaining activity.*n*.01 launches a happening.*n*.01 droid at the ORGANIZATION 1 7 0 which property.*n*.01 near the person.*n*.01 enlisted_person.*n*.01 rear skilled_worker.*n*.01 who uses a instrumentality.*n*.03 of happening.*n*.01 chemical.*n*.01 to dislodge the device.*n*.01

Table 3: Results from the event-to-sentence experiments.

Experiment	Perplexity	BLEU
Original Sentence → Original Sent.	704.815	0.0432
Original Words Event → Original Sent.	909.182	0.0015
Generalized Event → Generalized Sent.	**54.264**	**0.0496**
All Generalized Events → Gen. Sent.	55.312	0.0402

We also looked at whether *event2sentence* performance would be improved if we used multiple events per sentence (when possible) instead of the default single event per sentence. We leave genre out because we did not use genre numbers when testing the events with genre. (See Section 4.1 for more details.) In an end-to-end system, even if we were to use the events with genre numbers, they would not feed events with genre numbers into the *event2sentence* part of the system.

5.2 Results and Discussion

The results of our *event2sentence* experiments are shown in Table 3. As seen in Table 3, translating generalized events to partially generalized sentences achieves the best BLEU score. In the case of *event2sentence*, BLEU score makes more sense as a metric since the task is a translation task. Translating original word events to original sentences faired the worst of all experiments on perplexity and BLEU score. This is likely due to the difficulty in guessing how to fill in missing words from sentences.

The generalized event to generalized sentence did substantially better than the original words events in both perplexity and BLEU score. This is likely due to the fact that translation to partially generalized sentences is an easier problem than translation to original sentences; the network has to learn that certain tokens must appear in the output and there are some grammatical relations between words.

Table 4: Generalized Events and their Expected vs Generated Generalized Sentences.

Input (Generalized Event)	Expected Output	Generated Output
⟨<NE>0, contiguous_location-47.8-1, child.*n*.02, EmptyParameter ⟩	Meanwhile <NE>0 meets <NE>1 the child.*n*.02 of the social_group.*n*.01 liabilities.*n*.01 substance.*n*.01 substance.*n*.01 <NE>2	<NE>0 meets female.*n*.02 new lost causal_agent.*n*.01 and tells female.*n*.02 that female.*n*.02 is a person.*n*.01
⟨male.*n*.02, learn-14-2-1, EmptyParameter, EmptyParameter ⟩	male.*n*.02 learns that <NE>0 lost female.*n*.02 causal_agent.*n*.01 to a aircraft.*n*.01 sound.*n*.04 and became physically handicapped in a subsequent misfortune.*n*.01	male.*n*.02 learns female.*n*.02 genitor.*n*.01 died a long happening.*n*.01 ago and <NE>0 has sent female.*n*.02 to LOCATION
⟨<NE>0, calibratable_cos-45.6-1, EmptyParameter, feeling.*n*.01⟩	<NE>0 falls in feeling.*n*.01 with <NE>1 who brings opposition.*n*.02 to female.*n*.02 state.*n*.02	However <NE>0 has fallen in feeling.*n*.01 with a person.*n*.01 of male.*n*.02 person.*n*.01 <NE>1
⟨<NE>0, characterize-29.2-1, EmptyParameter, EmptyParameter⟩ ⟨male.*n*.02, fit-54.3, firearm.*n*.01, EmptyParameter⟩ ⟨female.*n*.02, hit-18.1-1, male.*n*.02, EmptyParameter⟩ ⟨female.*n*.02, force-59-1, EmptyParameter, self-propelled_vehicle.*n*.01⟩	When <NE>0 sees male.*n*.02 is carrying a concealed firearm.*n*.01 female.*n*.02 knocks male.*n*.02 down and drives off with male.*n*.02 self-propelled_vehicle.*n*.01	When <NE>0 sees male.*n*.02 using a cutter.*n*.06 female.*n*.02 knocks male.*n*.02 away and drives off in a self-propelled_vehicle.*n*.01

Table 4 shows some examples of translations from generalized events to partially generalized sentences. The table shows the generalized event, the expected (ground truth) output, and the generated output for comparison. To get a full sense of how the sentences would read, imagine adding character names and other details as if one were completing a *Mad-Libs* game.

Due to the nature of the generalized events, small specifics do not matter as long as the system is consistent within the story. For example, both “mother” and “father” might map to “genitor.n.01”, and even if we saw “mother” in the training data before it was generalized, it is okay to use “father” in the final story, as long as that selection was remembered.

6 FUTURE WORK

The question remains how to determine exactly what character names and noun details to use in place for the <NE>s and WordNet placeholders. In Figure 1, we propose the addition of Working Memory and Long-Term Memory modules. The Working Memory module would retain the character names and nouns in a lookup table that were removed during the eventification process. After a partially generalized sentence is produced by *event2sentence*, the system can use the Working Memory lookup table to fill character names and nouns back into the placeholders. The intuition is that from one event to the next, many of the details—especially character names—are likely to be reused.

In stories it is common to see a form of turn-taking between characters. For example the two events “John hits Andrew” followed by “Andrew runs away from John” followed by “John chases Andrew” illustrates the turn-taking pattern. If John were always used as the first named entity, the meaning of the example would be significantly altered. The continuous numbering of named entities (*event2event* experiment # 7) is designed to assist with maintaining turn taking patterns. Named entity tags would be consistent in the training data across event bigrams, allowing it to learn the pattern. As event bigrams are generated, it is possible for the alternating named entity pattern to manifest so that character names can be inserted from Working Memory.

There are times when the Working Memory will not be able to fill named entity and WordNet synset placeholder slots because the most recent event bigram does not contain the element necessary for reuse. The Long-Term Memory maintains a history of all named entities and nouns that have ever been used in the story and information about how long ago they were last used. See [13] for a cognitively plausible event-based memory that can be used to compute the salience of entities in a story. The story generation system would use the Long-Term Memory to find the most salient entity of the appropriate type to fill in slots that cannot be filled by the Working Memory. The underlying assumption is that stories are more likely to reuse existing entities and concepts than introduce new entities and concepts.

Our model of automated story generation as prediction of successor events is simplistic; it assumes that stories can be generated by a language model that captures generalized patterns of event co-occurrence. Story generation can also be formalized as a planning problem, taking into account communicative goals. In storytelling, a communicative goal can be to tell a story about a particular topic, to include a theme, or to end the story in a particular way (e.g., “happily ever after”). In future work, we plan to replace the *event2event* network with a reinforcement learning process that can perform lookahead to analyze whether potential successor events are likely to lead to communicative intent being met.

7 CONCLUSIONS

In automated story generation, event representation matters. We hypothesize that by using our intuitions into storytelling we can select a representation for story events that maintains semantic meaning of textual story data while reducing sparsity of events. The sparsity of events, in particular, results in poor story generation performance. Our experiments with different story representations during *event2event* generation back our hypothesis about event representation. We found that the events that abstract away from natural language text the most improve the generative ability of a recurrent neural network story generation process. Event bigrams did not significantly harm the generative model and will likely help with coherence as they incorporate more history into the process, although story coherence is difficult to measure and was not evaluated in our experiments.

Although generalization of events away from natural language appears to help with event successor generation, it poses the problem of making story content unreadable. We introduced a second neural network, *event2sentence*, that learns to translate generalized events back into natural language. This is important because it is possible for *event2event* to generate events that have never occurred (or has occurred rarely) in a story training corpus. By allowing *event2sentence* to generate partially generalized sentences instead of full sentences, we are able to achieve plausible, human-readable sentences. We present a proposed pipeline architecture for filling in missing details in automatically generated partially generalized sentences.

The pursuit of automated story generation is nearly as old as the field of artificial intelligence itself. Whereas prior efforts saw success with hand-authored domain knowledge, machine learning techniques and neural networks provide a path forward toward the vision of open story generation, the ability for a computational system to create stories about any conceivable topic without human intervention other than providing a comprehensive corpus of story texts.

ACKNOWLEDGMENTS

This work is supported by DARPA W911NF-15-C-0246. The views, opinions, and/or conclusions contained in this paper are those of the author and should not be interpreted as representing the official views or policies, either expressed or implied of the DARPA or the DoD. The authors would like to thank Murtaza Dhuliawala, Animesh Mehta, and Yuval Pinter for technical contributions.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. (2015). http://tensorflow.org/ Software available from tensorflow.org.

[2] David Bamman, Brendan O'Connor, and Noah A Smith. 2014. Learning latent personas of film characters. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. 352.

[3] Nathanael Chambers and Daniel Jurafsky. 2008. Unsupervised Learning of Narrative Event Chains.. In *ACL*, Vol. 94305. Citeseer, 789–797.

[4] David K. Elson and Kathleen R. McKeown. 2010. Automatic Attribution of Quoted Speech in Literary Narrative. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI'10)*. AAAI Press, 1013–1019. http://dl.acm.org/citation.cfm?id=2898607.2898769

[5] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, 363–370.

[6] Pablo Gervás, Belen Díaz-Agudo, Federico Peinado, and Raquel Hervás. 2005. Story Plot Generation based on CBR. *Journal of Knowledge-Based Systems* 18, 4–5 (2005), 235–242.

[7] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[8] Ahmed Khalifa, Gabriella AB Barros, and Julian Togelius. 2017. DeepTingle. *arXiv preprint arXiv:1705.03557* (2017).

[9] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*. 3294–3302.

[10] Michael Lebowitz. 1987. Planning stories. In *Proceedings of the 9th Annual Conference of the Cognitive Science Society*. 234–242.

[11] Boyang Li, Stephen Lee-Urban, George Johnston, and Mark O. Riedl. 2013. Story Generation with Crowdsourced Plot Graphs. In *27th AAAI Conference on Artificial Intelligence*. 598–604. http://www.cc.gatech.edu/

[12] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010

[13] Lara J. Martin, Brent Harrison, and Mark O. Riedl. 2016. Improvisational Computational Storytelling in Open Worlds. In *Ninth International Conference on Interactive Digital Storytelling (ICIDS 2015) / INT 9*, Vol. 10045. Springer International Publishing, 73–84.

[14] James R. Meehan. 1976. *The Metanovel: Writing Stories by Computers*. Ph.D. Dissertation. Yale University.

[15] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[16] George A. Miller. 1995. WordNet: A Lexical Database for English. *Commun. ACM* 38, 11 (1995), 39–41.

[17] Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. *arXiv preprint arXiv:1604.01696* (2016).

[18] Nasrin Mostafazadeh, Michael Roth, Annie Louis, Nathanael Chambers, and James F Allen. 2017. LSDSem 2017 Shared Task: The Story Cloze Test. *LSDSem 2017* (2017), 46.

[19] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 311–318.

[20] Rafael Pérez y Pérez and Mike Sharples. 2001. MEXICA: A computer model of a cognitive account of creative writing. *Journal of Experimental and Theoretical Artificial Intelligence* 13 (2001), 119–139.

[21] Karl Pichotta and Raymond J. Mooney. 2016. Learning Statistical Scripts with LSTM Recurrent Neural Networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*.

[22] Karl Pichotta and Raymond J. Mooney. 2016. Using Sentence-Level LSTM Language Models for Script Inference. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

[23] Julie Porteous and Marc Cavazza. 2009. *Controlling Narrative Generation with Planning Trajectories: The Role of Constraints*. Springer Berlin Heidelberg, Berlin, Heidelberg, 234–245. DOI: http://dx.doi.org/10.1007/978-3-642-10643-9_28

[24] Gerald Prince. 1987. *A Dictionary of Narratology*. University of Nebraska Press, Lincoln.

[25] Mark Owen Riedl and Robert Michael Young. 2010. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research* (2010).

[26] Melissa Roemmele, Sosuke Kobayashi, Naoya Inoue, and Andrew M Gordon. 2017. An rnn-based binary classifier for the story cloze test. *LSDSem 2017* (2017), 74.

[27] Roger Schank and Robert Abelson. 1977. Scripts. In *Scripts Plans Goals and Understanding*. Chapter 3, 36–68.

[28] Karin Kipper Schuler. 2005. VerbNet: A broad-coverage, comprehensive verb lexicon. (2005).

[29] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

[30] 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 (2012), 16:1–16:35.