The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Event Representations for Automated Story Generation with Deep Neural Nets

**Lara J. Martin, Prithviraj Ammanabrolu, Xinyu Wang,
William Hancock, Shruti Singh, Brent Harrison, Mark O. Riedl**
School of Interactive Computing
Georgia Institute of Technology
Atlanta, GA
{ljmartin; raj.ammanabrolu; lillywang1126; 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.

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 (Young et al. 2013) or case-based reasoning (Gervás et al. 2005). 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 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 (Li et al. 2013) is the problem of automatically generating a story about any domain without *a priori* manual knowledge engineering. Open story generation requires an intelligent system to either learn a domain model from available data (Li et al. 2013; Roemmele et al. 2017) or to reuse data and knowledge available from a corpus (Swanson and Gordon 2012).

In this paper, we explore the use of recurrent encoder-decoder neural networks (e.g., *Sequence-to-Sequence* (Sutskever, Vinyals, and Le 2014)) for open story generation. An encoder-decoder RNN 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 patterns observed in the training data. For open story generation, we must train the network on a dataset that encompasses as many domains as possible. For this work, we use a corpus of movie plot summaries extracted from Wikipedia (Bamman, O'Connor, and Smith 2014) under the premise that the set of movie plots on Wikipedia covers the range of domains that people want to tell stories about.

In narratological terms, an *event* is a unit of story featuring a world state change (Prince 1987). Textual story corpora, including Wikipedia movie plot corpora, are comprised of unstructured textual sentences. One benefit to dealing with movie plots is the clarity of events that occur, although this is often at 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 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 can 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. We measure the ability to predict the true successor event as an indicator of how event representations facilitate the modeling of context.

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. First, we discuss related work on automated story generation, followed by an introduction of our event representation. Then we introduce our *event2event* network and provide an evaluation of the event representations in the context of story generation. We show how the event representation can be used in *event2sentence* to generate human-readable sentences from events. We end with a discussion of future work and conclusions about these experiments and how our event representation and event2sentence model will fit into our final system.

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 (Young et al. 2013) or case-based reasoning using ontologies (Gervás et al. 2005). 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 *SayAnything* system (Swanson and Gordon 2012) uses textual case-based reasoning to identify relevant existing story content in online blogs. 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.

Recurrent neural networks can theoretically learn to predict the probability of the next character, word, or sentence in a story. Roemmele and Gordon (Roemmele et al. 2017) use a Long Short-Term Memory (LSTM) network (Hochreiter and Schmidhuber 1997) to generate stories. They use Skip-thought vectors (Kiros et al. 2015) to encode sentences and a technique similar to word2vec (Mikolov et al. 2013) to embedded entire sentences into 4,800-dimensional space. They trained their network on the BookCorpus dataset. Khalifa et al. (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*.

Based off of the theory of script learning (Schank and Abelson 1977), Chambers and Jurafsky (2008) learn causal chains that revolve around a protagonist. They developed a representation that took note of the event/verb 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 (2016a) 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 e_s, e_o, and e_p 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* (Mostafazadeh et al. 2016) 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 (Mostafazadeh et al. 2017).

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 from 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 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 (2016a), 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. Events are created by first extracting dependencies with Stanford's CoreNLP (Manning et al. 2014) and locating the appropriate dependencies mentioned above. If the object or modifier cannot be identified, we insert the placeholder $EmptyParameter$, which we will refer to as $\varnothing$ in this paper.

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 verbal or sentential conjunctions in the sentence. Consider the sentence "John and Mary went to the store," our algorithm would extract two events: $\langle john, go, store, \varnothing \rangle$ and $\langle mary, go, store, \varnothing \rangle$. The average number of events per sentence was 2.69.

Our experiments below used a corpus of movie plots from Wikipedia (Bamman, O'Connor, and Smith 2014), which we cleaned to remove any 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.

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 were identified (cf. (Finkel, Grenager, and Manning 2005)), and "PERSON" names were replaced with the tag <CHAR>n, where n indicates the n-th character name in the sentence. Other named entities were labeled as their named entity recognition (NER) category (e.g. LOCATION, ORGANIZATION, etc.). The rest of the nouns were replaced by the WordNet (Miller 1995) 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 (Schuler 2005) version 3.2.4[1] frames (e.g. "arrived"/"arriving" become "escape-51.1").
- **Character Name Numbering.** There were two ways of numbering the character names that we experimented with. One way had the character name numbering reset with every sentence (consistent within sentence)–or, *sentence CHARs*, our "default". The other way had the numbering reset after every input-output pair (i.e. consistent across two sentences)–or, *continued CHARs*.
- **Adding Genre Information.** We ran topic modeling on the entire corpus using Python's Latent Dirichlet Analysis[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 genre number which 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 (Pichotta and Mooney 2016a). The experiments in the next section show how different representations affect the ability of a recurrent neural network to predict story continuations.

Event2Event

The *event2event* network is a recurrent multi-layer encoder-decoder network based on (Sutskever, Vinyals, and Le 2014). 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 the n-th event, 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. 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:

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

where x is a token in the text, and

$$p(x) = \frac{count(x)}{\sum_{y \in Y} count(y)} \quad (2)$$

where Y is the vocabulary. 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 between the

[1] https://verbs.colorado.edu/vn3.2.4-test-uvi/index.php

[2] https://pypi.python.org/pypi/lda

generated and expected output where n varies from 1 to 4 (Papineni et al. 2002).

We use a greedy decoder to produce the final sequence by taking the token with the highest probability at each step.

$$\hat{W} = \arg\max_{w} Pr(w|S) \quad (3)$$

where $\hat{W}$ is the generated token appended to the hypothesis, S is the input sequence, and w represents the possible output tokens.

Experimental Setup

For each experiment, we trained a sequence-to-sequence recurrent neural net (Sutskever, Vinyals, and Le 2014) using Tensorflow (Abadi et al. 2015). Each network 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, 4-word event representation: $\langle s, v, o, m \rangle$ with no abstraction and using original character names.
2. *Original Words with <CHAR>s.* This experiment is identical to the previous except entity names that were classified as "PERSON" through NER were substituted with <CHAR>n.
3. *Generalized.* Using the same 4-word event structure, we replaced character names and generalized all other words through WordNet or VerbNet, following the procedure described earlier.

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 <CHAR>s.* This experiment mirrors the previous with the exception of the number of the <CHAR>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. The genre number was used in training for *event2event* but removed from inputs and outputs before testing; it artificially inflated BLEU scores since 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 <CHAR>s.* This experiment has the same continued <CHAR> numbering as experiment #4 had but with *event2event* trained on event bigrams.
8. *Generalized Bigram + Genre.* This is a combination of the ideas from experiments #5 and #6: generalized events in event bigrams and with genre added.

The following three 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.

9. *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 $n+1$.
10. *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.
11. *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: $e_j^1\ e_j^2$.

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 (2016b). Deleting words did little to improve the predictive ability of our *event2event* network. However, perplexity improved significantly once character names were replaced by generalized <CHAR> tags, followed by generalizing other words.

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 events with the original words (with or without character names) performed better in terms of BLEU score, it is our belief that BLEU is not the most appropriate metric for event 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 event 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. 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 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.

Event2Sentence

Unfortunately, events are not human-readable and must be converted to natural language sentences. Since 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. For example, the event ⟨relative.n.01, characterize-29.2, male.n.02, feeling.n.01⟩ could, hypothetically, have come from the sentence "Her brother praised the boy for his empathy." In actuality, this event came from the sentence "His uncle however regards him with disgust."

Table 1: Results from the event2event experiments. Best values from each of these three sections (baselines, additions, and multiple events) are bolded.

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

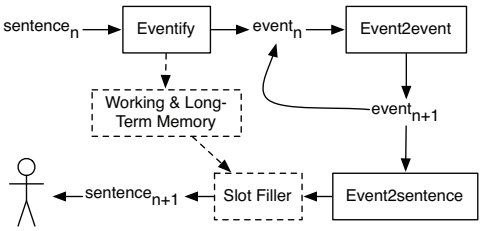

Figure 1: Our automated story generation pipeline. Dashed boxes and arrows represent future work.

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. Furthermore, our experiments with event representations for *event2event* indicate that greater generalization lends to better story generation. However, 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. 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 $e_{n+1} = \langle s^{n+1}, v^{n+1}, o^{n+1}, m^{n+1} \rangle$. As before, we use a recurrent encoder-decoder network based on (Sutskever, Vinyals, and Le 2014). 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 for filling in story specifics. 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 natural language sentences.

Experimental Setup

The setup for this set of experiments is almost identical to that of the *event2event* experiments, with the main difference being that we used PyTorch[3] which more easily lent itself to

[3]http://pytorch.org/

implementing beam search. The LSTM RNN networks in these experiments use beam search instead of greedy search to aid in finding a more optimal solution while decoding.

The beam search decoder works by maintaining a number of partial hypotheses at each step (known as the beam width or B, where B=5). Each of these hypotheses is a potential prefix of a sentence. At each step, the B tokens with the highest probabilities in the distribution are used to expand the partial hypotheses. This continues until the end-of-sentence tag is reached.

The input for these experiments was the events of a particular representation and the output was a newly-generated sentence based on the input event(s). 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. Examples of this can be seen in Table 3. However, we tested the *event2sentence* network on the same events that were used for the *event2event* experiments in order to conduct controlled experiments–we know the sentences from which they came–and compute perplexity and BLEU.

To test *event2sentence* with an event representation that used the original words is relatively straight forward. Experimenting on translating generalized events to natural language sentences was more challenging since we would be forcing the neural net to guess character names, other nouns, and verbs.

We devised an alternative approach for generalized *event2sentence* whereby sentences were first partially eventified. That is, we trained *event2sentence* on generalized sentences where the "PERSON" named entities were replaced by <CHAR> tags, other named entities were replaced by their NER category, and the remaining nouns were replaced by WordNet Synsets. 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.

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

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. Alternatively, we automatically split and prune (S+P) sentences; removing prepositional phrases, splitting sentential phrases on conjunctions, and, when it does not start with a pronoun (e.g. who), splitting S' (read: S-bar) from its original sentence and removing the first word. This would allow us to evaluate sentences that would have fewer (ideally one) events extracted from each. For example,

> Lenny begins to walk away but Sam insults him so he turns and fires, but the gun explodes in his hand.

becomes

> Lenny begins to walk away. Sam insults him. He turns and fires. The gun explodes.

Although splitting and pruning the sentences should bring most sentences down to a single event, this isn't always the case. Thus, we ran an *event2sentence* experiment where we extracted all of the events from the S+P sentences.

Table 2: Results from the event2sentence experiments.

Experiment	Perplexity	BLEU
Original Words Event → Original Sentence	1585.46	0.0016
Generalized Event → Generalized Sentence	56.516	0.0331
All Generalized Events → Gen. Sentence	59.106	0.0366
Original Words Event → S+P Sentence	490.010	**0.0764**
Generalized Event → S+P Gen. Sentence	**53.964**	0.0266
All Generalized Events → S+P Gen. Sent.	56.488	0.0283

Results and Discussion

The results of our *event2sentence* experiments are shown in Table 2. Although generalizing sentences improves perplexity drastically, splitting and pruning sentences yields better BLEU scores when the original words are kept. In the case of *event2sentence*, BLEU scores make more sense as a metric since the task is a translation task. Perplexity in these experiments appears to correspond to vocabulary size.

Generalized events with full-length generalized sentences have better BLEU scores than when the original words are used. However, when we work with S+P sentences, the pattern flips. We believe that because both S+P and word generalizing methods reduce sparsity of events, when they are combined too much information is lost and the neural network struggles to find any distinguishing patterns.

Table 3 shows examples from the entire pipeline as it currently exists, that is from one sentence to the next sentence without slot filling (See Figure 1). To get a full sense of how the generalized sentences would read, imagine adding character names and other details as if one were completing a *Mad-Libs* game.

Future Work

The question remains how to determine exactly what character names and noun details to use in place for the <CHAR>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 that were removed during the eventification process. The names and nouns could then be retrieved at a later time during the story. After a partially generalized

Table 3: End-to-end examples on previously-unseen input data. Let ∅ represent an empty (unfilled) parameter. Examples come from synopses of *Harry Potter and the Prisoner of Azkaban*, *Tales of Monkey Island* (with Guybrush changed to John), and *Carry On Regardless*, respectively.

Experiment	Input	Extracted Event(s)	Generated Next Event(s)	Generated Next Sentence
All Generalized Events & Generalized Sentence	He reaches out to Remus Lupin, a Defence Against the Dark Arts teacher who is eventually revealed to be a werewolf.	⟨male.n.02, get-13.5.1, ∅, <CHAR>0⟩ ⟨ORGANIZATION, say-37.7-1, monster.n.01, ∅⟩	⟨monster.n.01, amuse-31.1, sarge, ∅⟩ ⟨monster.n.01, amuse-31.1, realize, ∅⟩ ⟨monster.n.01, conjecture-29.5-1, ∅, ∅⟩ ⟨male.n.02, conduit.n.01, entity.n.01, ∅⟩ ⟨male.n.02, free-80-1, ∅, penal_institution.n.01⟩	When monster.n.01 nemesis.n.01 describes who finally realizes male.n.02 can not, dangerous entity.n.01 male.n.02 is released from penal_institution.n.01.
Generalized Event & Generalized S+P Sentence	John unwittingly unleashes an insidious pox that rapidly spreads across the Caribbean.	⟨<CHAR>0, disassemble-23.3, contagious_disease.n.01, ∅⟩	⟨male.n.02, spatial_configuration-47.6, ∅, adopt-93 ⟩	male.n.02 crumples and is about to be sheath.n.02.
Original Words Event & S+P Sentence	He thinks he's on a top secret spying mission.	⟨he, think, ∅, ∅⟩	⟨she, come, know, truth⟩	She has come to the truth.

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" & "Andrew runs away from John" followed by "John chases Andrew" illustrate the turn-taking pattern. If John was always used as the first character name, the meaning of the example would be significantly altered. The continuous numbering of character names (*event2event* experiment #7) is designed to assist event bigrams with maintaining turn-taking patterns.

There are times when the Working Memory will not be able to fill character name 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 character names and nouns that have ever been used in the story and information about how long ago they were last used. See Martin et al. (2016) for a cognitively-plausible event-based memory that can be used to compute the salience of entities in a story. The underlying assumption is that stories are more likely to reuse existing entities and concepts than introduce new ones.

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 domain, to include a theme, or to end the story in a particular way. 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.

Conclusions

In automated story generation, event representation matters. We hypothesize that by using our intuitions about 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 support our hypothesis about event representation. We found that the events that most abstract away from natural language text 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 events with generalized or original words back into natural language. This is important because it is possible for *event2event* to generate events that have never occurred (or have occurred rarely) in a story training corpus. We maintain that being able to recover human-readable sentences from generalized events is valuable since our *event2event* experiments show use that they are preferred, and it is necessary to be able to fill in specifics later for dynamic storytelling. 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 domain 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 authors 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

Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Goodfellow, I.; Harp, A.; Irving, G.; Isard, M.; Jia, Y.; Jozefowicz, R.; Kaiser, L.; Kudlur, M.; Levenberg, J.; Mané, D.; Monga, R.; Moore, S.; Murray, D.; Olah, C.; Schuster, M.; Shlens, J.; Steiner, B.; Sutskever, I.; Talwar, K.; Tucker, P.; Vanhoucke, V.; Vasudevan, V.; Viégas, F.; Vinyals, O.; Warden, P.; Wattenberg, M.; Wicke, M.; Yu, Y.; and Zheng, X. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Bamman, D.; O'Connor, B.; and Smith, N. A. 2014. Learning latent personas of film characters. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 352.

Chambers, N., and Jurafsky, D. 2008. Unsupervised learning of narrative event chains. In *ACL*, volume 94305, 789–797. Citeseer.

Finkel, J. R.; Grenager, T.; and Manning, C. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, 363–370. Association for Computational Linguistics.

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*.

Kiros, R.; Zhu, Y.; Salakhutdinov, R. R.; Zemel, R.; Urtasun, R.; Torralba, A.; and Fidler, S. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, 3294–3302.

Li, B.; Lee-Urban, S.; Johnston, G.; and Riedl, M. O. 2013. Story Generation with Crowdsourced Plot Graphs. In *27th AAAI Conference on Artificial Intelligence*, number Weyhrauch 1997, 598–604.

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.

Martin, L. J.; Harrison, B.; and Riedl, M. O. 2016. Improvisational computational storytelling in open worlds. In *Ninth International Conference on Interactive Digital Storytelling (ICIDS 2015) / INT 9*, volume 10045, 73–84. Springer International Publishing.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

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

Mostafazadeh, N.; Chambers, N.; He, X.; Parikh, D.; Batra, D.; Vanderwende, L.; Kohli, P.; and Allen, J. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. *arXiv preprint arXiv:1604.01696*.

Mostafazadeh, N.; Roth, M.; Louis, A.; Chambers, N.; and Allen, J. F. 2017. Lsdsem 2017 shared task: The story cloze test. *LSDSem 2017* 46.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, 311–318. Association for Computational Linguistics.

Pichotta, K., and Mooney, R. J. 2016a. Learning Statistical Scripts with LSTM Recurrent Neural Networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*.

Pichotta, K., and Mooney, R. J. 2016b. Using Sentence-Level LSTM Language Models for Script Inference. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Prince, G. 1987. *A Dictionary of Narratology*. Lincoln: University of Nebraska Press.

Roemmele, M.; Kobayashi, S.; Inoue, N.; and Gordon, A. M. 2017. An rnn-based binary classifier for the story cloze test. *LSDSem 2017* 74.

Schank, R., and Abelson, R. 1977. Scripts. In *Scripts Plans Goals and Understanding*. chapter 3, 36–68.

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

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.

Young, R. M.; Ware, S. G.; Cassell, B. A.; and Robertson, J. 2013. Plans and planning in narrative generation: a review of plan-based approaches to the generation of story, discourse and interactivity in narratives. *Sprache und Datenverarbeitung, Special Issue on Formal and Computational Models of Narrative* 37(1-2):41–64.