

GraphPlan: Story Generation by Planning with Event Graph

Hong Chen^{1,3}, Raphael Shu¹, Hiroya Takamura^{2,3}, Hideki Nakayama¹

¹The University of Tokyo, ²Tokyo Institute of Technology,

³National Institute of Advanced Industrial Science and Technology, Japan

{chen, nakayama}@nlab.ci.i.u-tokyo.ac.jp, raphael@uaca.com, takamura.hiroya@aist.go.jp

Abstract

Story generation is a task that aims to automatically produce multiple sentences to make up a meaningful story. This task is challenging because it requires high-level understanding of semantic meaning of sentences and causality of story events. Naive sequence-to-sequence models generally fail to acquire such knowledge, as the logical correctness can hardly be guaranteed in a text generation model without the strategic planning. In this paper, we focus on planning a sequence of events assisted by event graphs, and use the events to guide the generator. Instead of using a sequence-to-sequence model to output a storyline as in some existing works, we propose to generate an event sequence by walking on an event graph. The event graphs are built automatically based on the corpus. To evaluate the proposed approach, we conduct human evaluation both on event planning and story generation. Based on large-scale human annotation results, our proposed approach is shown to produce more logically correct event sequences and stories.

1 Introduction

Narrative Intelligence (Mateas and Sengers 2003) is one form of Humanistic Artificial Intelligence that requires the system to organize, comprehend, and reason about narratives and produce meaningful responses. Story generation tasks can be considered as a test bed for examining whether a system develops a good understanding of the narratives.

Other than leaving the model to output random sentences, the model is usually given a specific topic (e.g., title or prompt) or visual information (e.g., image or video). One straight-forward approach for these story generation tasks is to leverage a sequence-to-sequence model to predict sentences sequentially. Although the model can be trained to capture the word-prediction distribution from training data, it has two serious drawbacks when applied to generate stories: 1) Conditional Language model (i.e., decoder) tends to assign high probabilities to generic, repetitive words, especially when beam search is applied in the decoding phase (Holtzman et al. 2019); 2) Sequence-to-sequence models often fail to produce logically correct stories.

Recently, huge interests have been aroused to decompose story generation into two phases: planning and generation (Yao et al. 2019; Goldfarb-Tarrant, Feng, and Peng 2019; Xu et al. 2018; Fan, Lewis, and Dauphin 2019). Plan-

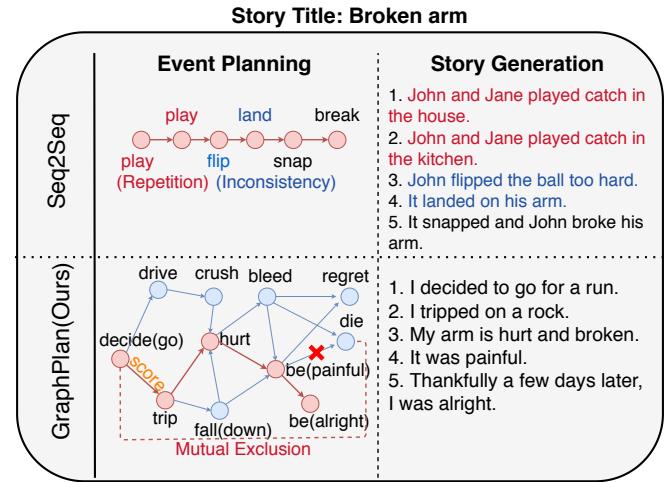


Figure 1: Comparison between sequence-to-sequence model and GraphPlan (ours). Two problems in the sequence-to-sequence model when generating events: **Repetition** and **Logical Inconsistency**. Repeated words (e.g., play) in the storyline result in the repeated sentences in the generated stories. Besides, The logic between “land” and “snap” lacks causality, thus generating incoherent stories. On the contrary, our GraphPlan method does not rely on any language model, it applies beam search on the event graph based on a well-designed score function. The mutually exclusive set further ensures the global logical consistency for the planned events.

ning (Meehan 1976; Riedl and Young 2010) creates a high-level abstraction or a blueprint to encourage the generator to focus on the flow of a story, similar to making an outline before writing. The planned elements are referred to as *events* in many papers. However, the detailed definition of events varies. For instance, an event can be represented as a verb argument pair (e.g., *(admits, subj)*) (Chambers and Jurafsky 2008), a tuple of subject, verb, object and modifier or “wildcard” (e.g., *(PERSON0, correspond-36.1, empty, PERSON1)*) (Martin et al. 2018; Ammanabrolu et al. 2019) or a reconstructed verb phrase (e.g., *decide(go)*) (Peng and Roth 2016). In this paper, we follow Peng and Roth (2016) to rep-

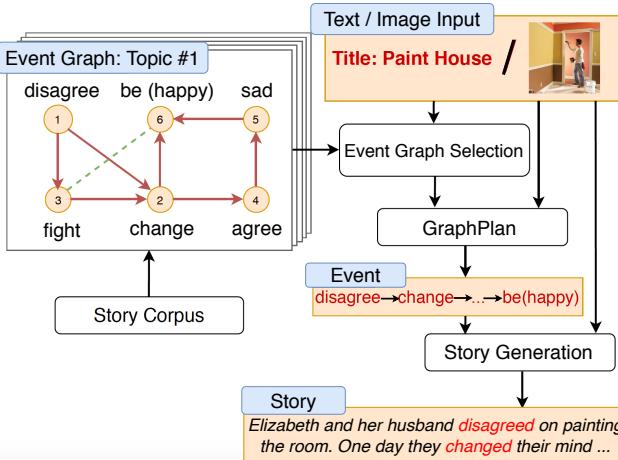


Figure 2: Overview of our approach. In the preprocessing step, we cluster the stories into K topics and build an event graph for each topic. In the planning step, a event graph selection module select a event graph based on the input. Then a related event graph is retrieved. The event planning model generates a sequence of events. Finally, based on the input and the planned events, a story generation module generates the story. The dash line denotes the mutually exclusive events that can hardly coexist in one storyline.

resent an event with verb phrases.

Previous works (Yao et al. 2019) have shown that if the events are well-planned, then the correctness of generated stories is almost guaranteed, and furthermore, the stories can be easily controlled by modifying the events. However, existing approaches (Meehan 1976; Goldfarb-Tarrant, Feng, and Peng 2019; Martin et al. 2018; Ammanabrolu et al. 2019) regard event generation as an abstracted version of story generation. In other words, they treat each event as one token and use a sequence-to-sequence model to make a plan of the events. Our preliminary experiments show that repetition and logical inconsistency problems happen in the event sequence and same problems occur in the generated stories. Figure 1 shows an example using sequence-to-sequence in event planning. We can see that the both events and generated stories are repeated and illogical.

In this paper, instead of leveraging a sequence-to-sequence model on event planning, we propose a planning method **GraphPlan**. To plan the events, GraphPlan walks on a topic-specific event graph with beam search. Event graphs are adopted for story generation even before the emergence of neural-based models (Weyhrauch 1997; Chen et al. 2009; Regneri, Koller, and Pinkal 2010; McIntyre and Lapata 2010; Li et al. 2013). An event graph represents the logical flow of events based on the facts presented in a corpus. With a learned event graph, we can walk on it and produce a reasonable event sequence. We follow the graph setting in Li et al. (2013), in which each graph is composed of event nodes, connections and a set of mutually exclusive events.

To generate a story, we first identify the topic based on

the input (e.g., title or image) and retrieve a related event graph. We then plan the events by running beam search with a score function that takes the event-event coherence and input-event coherence into account. Finally, a *story generation* module transforms the planned event sequence into a readable story. Figure 1 shows an example of using GraphPlan and Figure 2 depicts the whole pipeline of our proposed approach.

We conduct experiments on open story generation to evaluate how event graphs benefit the task. Our approach is shown to significantly outperform baseline models that generate events with sequence-to-sequence models in terms of logical consistency. We also conduct Story Cloze Test to further validate the effectiveness of the event graphs and the mutually exclusive sets. Our contributions can be summarized as follows:

- We propose a score-based beam search approach to plan story events with an event graph.
- Comparing to baseline models, our graph-based planning approach results in much better logical correctness in story generation tasks according to the human evaluation.
- Experiments on Story Cloze Test directly confirms the high accuracy of the proposed event planning approach.

2 Related Work

Planning for Story Generation Several approaches have been explored to plan a skeleton of the story before actual generation. Before the emergence of neural-based models, Reiter and Dale (1997) and Riedl (2010) attempted to use hand crafted rules to arrange actions into character sequences. Recently, with the help of neural sequence-to-sequence models, Xu et al. (2018) proposed to generate multiple key phrases and expand them into a complete story. A built-in key phrases generation module is used in their model architecture. In contrast to Xu et al. (2018), some works explicitly plan a sequence of events (Martin et al. 2018; Ammanabrolu et al. 2019; Tambwekar et al. 2019), keywords (Yao et al. 2019; Ippolito et al. 2019) or actions (Fan, Lewis, and Dauphin 2019) before generating the story based on the planned items.

All of these planning models rely on a language model for planning without following an external structure of events, which resulted in degraded performance (Holtzman et al. 2019). Compared with these works, the main contribution of this paper is to propose a planning method based on automatically created event graphs. Instead of a language model, we use score-based beam search to generate a sequence of events by walking on the graph.

Graph-based Story Planning Event graph is a variant of plot graph whose nodes represent events. A quantity of research made progress on generating stories from plot graphs (Weyhrauch 1997; Chen et al. 2009; Regneri, Koller, and Pinkal 2010; McIntyre and Lapata 2010; Li et al. 2019). Li et al. (2013) proposed a plot graph on story generation tasks which is most related to our work. They crowd-sourced the story corpus and manually created plot nodes and edges

in the graph. In their graph, mutually exclusive events are not allowed to be present in the same story.

In this work, both the event graphs and mutually exclusive sets are automatically generated. We further propose an event planning method taking into account the relations between events and various inputs (i.e., title or image).

3 Event Graph Construction

As a preprocessing step, we first extract events automatically from a corpus. Then, we divide the corpus into several topics. Finally, we build an event graph for each topic.

Data-based Event Extraction Following Peng and Roth (2016), we represent each event with a verb phrase. Unlike other representations, a verb phrase is the minimum unit in one sentence which is abstract, simple and comprehensible for humans. From our observation, this representation does not have a severe sparseness problem. In statistics, each event can connect over 3 next possible events on average. Please note that our work does not investigate event representation, we focus on planning a more logical event sequence. Specifically, as a preprocessing step, we parse all sentences with semantic role labeling and extract the verb phrases. If an extracted verb has an argument with semantic role “AM-NEG” (negation) for a verb, we add (*not*) before it (e.g., (*not*)*take*). If a verb is followed by a preposition, we append the prepositional word to the verb (e.g., *take(over)*). If the label is “AM-PRD” (secondary predicate), we make an event from it (e.g., *be(excite)*). Finally, if two verbs are close to each other within five-word distance in the corpus, we combine them to make an event (e.g., *decide(buy)*). All words in the event are stemmed with NLTK (Bird, Klein, and Loper 2009).

Topic Modelling Generally, a story dataset contains a variety of topics ranging from animal, health, to robbery. Here, we use Latent Dirichlet Allocation (LDA) (Blei, Ng, and Jordan 2003) to infer the topics in the corpus. Considering that the relation between events drastically changes according to the topic, in this work, we build an independent event graph for each topic. Formally, we denote e_1^k, \dots, e_t^k the event set from the stories that belong to the k-th topic \mathcal{T}^k in the corpus. These events would be used as nodes for the event graph of \mathcal{T}^k . LDA clusters the stories and thus reduces the amount of unique events in each graph, which will make the graph walking algorithm more efficient.

Event Connection After collecting the events from a corpus for each topic, we need to find connections among these events to build a graph. The connections are represented as directed edges whose direction indicates possible next events. In practice, if events e_i and e_j occur adjacently in the text, we add an edge $e_i \rightarrow e_j$. An example of an event graph can be found in Figure 2.

Mutually Exclusive Set Following the graph setting in Li et al. (2013), there are events (e.g. “die” and “be(happy)”) that are mutually exclusive and cannot be placed in one story. These mutually exclusive relations are considered as exceptions and difficult to be represented along with the event graph. We create a held-out set consisting of mutually exclusive event pairs for each graph.

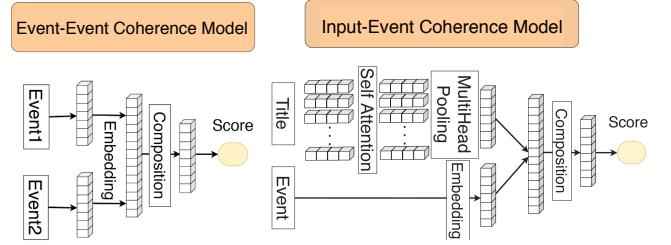


Figure 3: Coherence models we used in this paper. The event-event coherence model outputs a coherence score for two events. The input-event coherence model takes a title and an event as input. Both coherence models finally produce a score within 0 to 1. These coherence scores decide the next event when running beam search.

To identify these mutually exclusive events from the constructed graphs, we prepare an event-event coherence model to detect the coherence score between two events. We prevent two events with low coherence scores from coexisting in the planned events. The model architecture is based on compositional neural networks (Granroth-Wilding and Clark 2016), as shown in Figure 3. The model takes two events (e_i, e_j) represented with unique embeddings and outputs a coherence score normalized with the sigmoid function $f_{event}(e_i, e_j) \in [0, 1]$. We use contrastive training to optimize the model. Here, positive examples are the events extracted from the same story or title, whereas negative examples are randomly sampled from the events in different stories. Let (e_i, e_j) denote a positive pair of events and \tilde{e}_j denote a randomly sampled event. The training loss for the event-event coherence model is defined as:

$$L_{event} = \max(0, -f_{event}(e_1, e_2) + f_{event}(e_1, \tilde{e}_2) + m) \quad (1)$$

where m is a fixed margin. Finally, we consider two events are mutually exclusive if the coherence score falls below a certain threshold τ .

On average, after taking the mutually exclusive sets into account, each event graph can still plan over one million different possible event sequences. Please refer to the supplementary materials for more statistics details of event graphs. Additionally, these in-topic event graphs can be hierarchically combined into a larger graph if the model is required to generate longer discourse-level stories. This will be a future direction of our work.

4 GraphPlan: Planned Story Generation with Event Graph

In this section, we describe our approach for planned story generation. We separated the whole pipeline into two steps: 1) Our GraphPlan walks on the event graph and produces

a sequence of events as a blueprint of the story. 2) The story generation module then finalizes the texts following the planned events.

4.1 GraphPlan

Till now, each topic has a corresponding event graph. Before story generation, we propose GraphPlan to plan event sequences from the event graph. These planned events will be used to guide the story generation module in the next step. GraphPlan contains two steps. 1) Selecting an event graph for the input (i.e. title or image). 2) Generating an event sequence by walking on the graph.

Event Graph Selection Firstly, we identify the topic of inputs to retrieve the corresponding event graph. Depending on tasks, the inputs can be titles for open story generation, or images for the visual storytelling task. If the input is a piece of text, we directly use the LDA model we trained earlier to identify the topic.

Event Sequence Generation Once we identify the topic of input, we walk on the corresponding event graph to generate a sequence of events. In our experiments, we found that an autoregressive language model tends to produce repetitions, thus resulting in degraded performance. Hence, we propose to use a score-based generation method. The algorithm can be seen as a type of beam search, in which the candidate event sequences are ranked by a score function. Starting from a random event e_1 , we progressively search for the next event e_t in the following candidate set:

$$\{e_t \mid e_t \in \text{Graph}(e_{t-1}), \\ e_t \notin \text{Exclusive}(e_1, \dots, e_{t-1})\}, \quad (2)$$

where $\text{Graph}(\cdot)$ returns a set of possible next events in the graph, $\text{Exclusive}(\cdot)$ returns a set of mutually exclusive events. This filtering step greatly reduces the number of candidate events to consider.

To select the event from the candidate set, we rank all remaining candidate events with the following score function:

$$\text{Score}(e_t) = \frac{1}{\sum_{i=0}^{t-1} \lambda^i} \sum_{i=0}^{t-1} \lambda^i \log f_{\text{event}}(e_i, e_t) \\ + \log f_{\text{input}}(x, e_t) \quad (3)$$

where the first term of score function sums the event-event coherence score of candidate event e_t to each partially generated event e_i and gives more weights to recent events. λ denotes the decay rate. Then a decayed average applied over the score. The model used in producing the event-event coherence is the same model used in detecting mutually exclusive events. The second term is an input-event coherence score $f_{\text{input}}(x, e_t)$, which indicates the coherence score between event e_t and the input x . We propose an input-event coherence model to compute this score. Please refer to Figure 3 for details of parameterization. For the task of open story generation, the input-event coherence model is implemented with compositional neural networks, where the input x in Equation (3) is the title.

As a common practice for beam search, we set a budget of the number of candidates to explore (i.e., beam size). The candidates with the highest scores are maintained in the beam. The final candidate with the highest score is selected as the result of planning.

4.2 Story Generation Module

The generated event sequence will be sent to a story generation module, which converts the events into a story. This story generation module can be any type of model. Recently, large pre-trained language models show great capability of generating knowledgeable and informative sentences. Taking these advantages, the planned events are more likely to be logically connected in the generated story. Therefore, we apply GPT2-small (Radford et al. 2019) as our story generation module. During the training, we feed the module with the title words and events. A special token “<EOT>” separates the title and the events and another special token “<SEP>” is placed in every interval of the events. “<|endofinput|>” token is added at the end of the input. Besides, we also train an RNN-based sequence-to-sequence model that is fed with the same inputs for comparison.

However, as stated in Yao et al. (2019); Tan et al. (2020), Exposure bias problem happens when plan-write strategy is applied. To mitigate this problem, we alternatively add two kinds of noises into the inputs. 1) Mask 20% events with a “[MASK]” token. 2) Mask all events. The first noise encourages the model to generate sentences referring to all planned events. While, the second noise promotes the model to generate more related stories to the title. The effectiveness of two noises are analyzed in the supplementary material.

5 Experiment

5.1 Experiment Settings

We design two experiments to explicitly evaluate event quality and story quality. Firstly, we calculate the diversity score and conduct human evaluation on the planned events. Secondly, we use the story generation module to transform the events into full stories and conduct human evaluation to evaluate the story quality. Moreover, to further verify the correctness of our GraphPlan, we conduct experiments on Story Cloze Test. The details of model implementations for all experiments can be found in the supplementary material.

5.2 Dataset

ROCStories Corpora (Mostafazadeh et al. 2017) consists of 98,162 training stories and 1,874 stories for validation and testing. Each story contains a title which we use as the input and a five-sentence story as the target. Since titles are annotated only for training data, we split this training set into 8:1:1 for training, validating and testing. We applied clustering to the training split (i.e., 8 of 8:1:1) and obtained 500 topics, in which each topic represents one specific domain. Each story is generated from one specific domain in the following experiments. Gold event sequences that are used in planning methods are extracted from the stories in the corpus.

Diversity	S2S	S2S(R)	GP	GOLD
Dist-1	10.17%	11.35%	20.54%	24.92%
Dist-2	56.55%	58.92%	78.12%	87.75%

Table 1: Diversity of planned events. We can see that both sequence-to-sequence models achieve low diversity, while GraphPlan can achieve high diversity.

5.3 Baseline

S2S Following Yao et al. (2019), we use a sequence-to-sequence model (Bahdanau, Cho, and Bengio 2015), which straightforwardly generates events given the input titles.

S2S(R) To build a more competitive baseline, we adopt reward shaping in the sequence-to-sequence model. Like (Tambwekar et al. 2018), we apply policy gradient on

$$\begin{aligned} \nabla_{\theta} J(\theta) &= R(e_i) \nabla_{\theta} \log P(e_i | e_1, \dots, e_{i-1}; \theta) \\ R(v) &= \alpha \times r_1(v) \times r_2(v) \\ r_1(v) &= \log f_{\text{input}}(x, v) \\ r_2(v) &= \frac{\sum_{e \in E \wedge e \neq v} \log f_{\text{event}}(e, v)}{N - 1} \end{aligned} \quad (4)$$

where e denotes set of the events in the planning sequence, E denotes the events in the story, N denotes the number of events in the story, x denotes the input title and α denotes the normalization constant across the events in each training sample. During training, the gradient from e_i is multiplied by the Reward $R(e_i)$ that are proportional to $r_1(e_i)$ and $r_2(e_i)$. In briefly, $r_1(e_i)$ get larger when e_i is more related to the input x , while $r_2(e_i)$ become larger when e_i is more likely to coexist with all events $\{e | e \in E \wedge e \neq e_i\}$. This method enforces the model to focus on the event that has a high coherence score to the input and events in each training sample.

GR In this method, we apply random walk on the event graphs while considering mutually exclusive sets. We aims to show the importance of using coherence models by comparing with this method.

GP(Ours) This is our proposed method that plans events on an event graph with mutually exclusive set and coherence models.

5.4 Event Quality

We plan the events on randomly selected 1000 test data with different baselines and our proposal. We first test the diversity of generated sequences. We calculate Distinct-1 and Distinct-2 scores to show the percentage of unique unigram and bigram events in the whole generated events. Table 1 shows that the sequence-to-sequence model suffers from producing repeated unigram and bigram events. Graph plan produ more events in the full event set (more unigrams) and produces more combinations between events (more bigrams).

To further evaluate the quality of planned events, we conduct human evaluation. Instead of using overly abstract event representation as (Tambwekar et al. 2019), we use the verb phrase which is more comprehensive for humans. Thus,

Choices(%)	GP vs S2S		GP vs S2S(R)		GP vs GR	
	GP	S2S	GP	S2S(R)	GP	GR
Revelence	47.0	17.8	52.0	26.0	56.3	12.9
Logical	53.5	14.9	55.2	26.0	51.5	17.8

Table 2: Human evaluation on event planning. Cohen’s Kappa coefficient (κ) for all annotations are in the Moderate agreement (0.4–0.6). Sign tests further show the significant difference. p-values are < 0.01 for all pairwise comparisons.

Title	Married too fast
S2S	be → be → be → fall → marry
S2S(R)	be → go(up) → ask → say → marry
GR	want(do) → go → sit → wonder → call
GP	feel → decide → begin → start → regret
Title	New glasses
S2S	sit → be(unhappy) → have → go → find
S2S(R)	break → need → go → get → be(glad)
GR	wake(up) → be → (not)care → take → make
GP	buy → wear → break → shatter → decide(buy)
Title	The new dress
S2S	want → find → decide(make) → buy → have
S2S(R)	skip → go → have → let → be(black) → make
GR	celebrate → wear → look → pick(out) → wear → make
GP	want(dress) → want(change) → dress → wear → feel(beautiful)
Title	Grilled cheese
S2S	love → be → decide → forget → end(up)
S2S(R)	make → get → go → go → look → see
GR	feel(comfortable) → like → smile → decide → feel(full)
GP	melt → put → decide(roast) → burn → taste

Table 3: Examples of the planned events.

we request the annotators to compare the event sequences by two criteria: Relevance and Logicality. Table 2 shows the human evaluation results. From the results, we can see that our planned events (i.e., verb phrases) are more related to the input title and can be easily transformed into a story.

Table 3 shows some of the examples generated by different methods. The results show that sequence-to-sequence models tend to generate repetitive events. Specifically, it tends to output the event that occurs with high frequency in the corpus, such as “be”(there is sth.) and “go” (sb. go somewhere). This is common for a model trained under the framework of maximum likelihood estimation method. Although reward shaping (S2S(R)) helps a lot, the problem is still not eliminated. Without the limitation of coherence score, GR walks on graphs randomly to produce a sequence. As the graph is relatively not small, achieving a good event sequence is extremely challenging. We can see that our proposal GP produces more logical and diverse events, which humans can easily tell a story based on these given events.

5.5 Open Story Generation

The ultimate goal of event planning is to generate more relative and logical coherent stories. Human evaluation on the event sequence is subjective and tricky since the event is highly abstract. To prove that better event planning improves

Choices(%)	GP vs GPT2		GP vs S2S		GP vs S2S(R)		GP vs GR		GP vs GP+RNN	
	GP	GPT2	GP	S2S	GP	S2S(R)	GP	GR	GP	GP+RNN
Revelence	33.3	41.6	38.4	28.3	40.8	20.3	67.8**	23.2	37.2*	23.3
Interestingness	47.9	41.6	40.1	30.8	43.2	34.0	60.7*	39.3	44.2**	24.5
Logicality	64.6**	27.1	42.6**	19.5	44.6*	25.0	62.5**	33.9	37.2	32.6
Overall	56.3*	35.4	42.6**	21.6	42.3*	24.6	64.3**	30.4	46.5	37.2

Table 4: Human evaluation on open story generation. (+GPT2) are omitted for all methods except for GP+RNN. We calculate Cohen’s Kappa coefficient (κ). They are in the Moderate agreement (0.4-0.6) and a few of them are in the Fair agreement (0.2-0.4). We also conduct sign tests to show the significant difference as well. (*) denotes p-value is < 0.05 and (**) denotes p-value is < 0.01 . The result shows that our method achieves significant improvement on logicality, thereby achieving better overall performance.

Title: Fire next door	
GPT-2	<i>After the fire, the fire was out. The family who lived next door started to panic, so they followed it. [The firehouse across the street was still burning. The fire house was still burning next door.] (Repetition)</i>
S2S	make → think → go → see → get → apologize <i>Lisa’s new neighbor made a big fire outside his house. Lisa thought she would be sorry. She went outside to check on him. She saw that he was not there. Lisa’s neighbor could not get a word with him and apologized.</i>
S2S(R)	try → tell → leave → throw → take <i>Sally tried to fight a burning bush. The homeowner told her she couldn’t do it. She left a piece of wood in the bush next door. A woman threw the wood at sally. Sally took the wood to the fire department.</i>
GR	quit → be → look(buy) → find → go <i>I quit my job when i was 16. The only thing I ever didn’t do was take care of my children. Now I’m looking to buy a car. I found a cheap car on ebay. Now i can go out in the sun every day.</i>
GP+RNN	catch(on) → come → go(off) → evacuate → extinguish <i>The fire alarm was caught on. The firefighters came to help. The fire alarm went off. Everyone evacuated. The firemen extinguished it before it could go off.</i>
GP	catch(on) → come → go(off) → evacuate → extinguish <i>The house next door caught on fire. The fire department came to the scene. The fire alarm went off. The entire neighborhood evacuated. The fire department extinguished the fire.</i>
Gold	John woke up smelling like something was burning. He went outside. He saw the fire next door. He called the authorities. The firemen came to put out the fire.

Table 5: Examples of open story generation. The red word represents the events.

the story quality, we generate stories using the planned events and conduct human evaluation to assess the Relevance, Logicality, Interestingness and Overall scores. We use story generation module (i.e. GPT2 and RNN) to transform these planned events into the full stories.

We compare the following methods in this experiment.

GPT2. It is a large scale language model that shows great performance in generating stories in recent research. In this method we directly input the title to the GPT2 and generate the full stories.

*+GPT2. We associate the aforementioned event plan-

ning methods with GPT2 which is used in the story generation module. Thus, we compare with S2S+GPT2, S2S(R)+GPT2, GR+GPT2 and GP+GPT2.

GP+RNN. In this method, we use an RNN based sequence-to-sequence model to generate the full story which takes the title and the events as inputs. We compare this method to GP+GPT2 to show the effectiveness of large scale language models in transforming the events into the stories.

We conduct human evaluation on Amazon Mechanical Turk (AMT) over four aspects: *Relevance* (whether the story is related to the topic), *Interestingness* (whether the story content and style are interesting), *Logicality* (whether the story is logical), and *Overall* (overall quality). The full details of human evaluation are listed in the supplementary materials. We randomly sample 300 titles from the testing set and generate the stories via each method. Pairwise comparison is conducted to each criteria and each sample is assigned for two different workers to avoid randomness or personal bias. Table 4 shows that our approach performs better in Logicality, and Overall. Especially, our method greatly outperforms other planning methods in the Logicality measure, which suggests that our planned events are logically sound. We believe that two factors are the primary reasons of improved logic: 1) each event graph is built from the corpus and thus walking on the graph remains the events’ logical relations, and 2) The coherence models filter the candidates and the mutually exclusive set further eliminates the non-logical combinations when planning the events. Table 5 shows an example of stories generated by these methods. We show both the planned events and the stories. Directly using GPT2 produces repeated sentences. Both equipped with an auto-regressive model for event planning, the events planned by S2S and S2S(R) fail to output satisfactory results and thus results in the low logicality score in the generated sentences. Since there is no restriction on the event selection in GR, the event produced could be irrelevant to the title and even mutually contradictory. By using our proposed method, GP can plan a reasonable set of events and thus generate the most logical story.

5.6 Story Cloze Test

To better validate the effectiveness of our event graphs and mutual exclusive relation between events, we conduct Story Cloze Test. This task aims to select the right ending sentence from two candidates. We incorporate the event fea-

ACC(%)	Test v1.0	Test v1.5
DN	77.60	64.45
DN+Origin	78.87	67.64
DN+RandomWalk	79.36	68.09
DN+GraphPlan	80.15	69.45

Table 6: Results on story cloze test. DN, denote the DiffNet. From the results, events planned by our event graphs and mutually exclusive sets have positive effects on this task.

ture generated by different methods into the story cloze test. The accuracy of Story Cloze Test would reflect the quality of the event features. The event feature is learned by a mask language model (MLM) (i.e., BERT model with fewer parameters) (Devlin et al. 2018): if the training event sequence is more logical and reasonable, the learned feature by MLM would better fit the story cloze test. To prove that our event graph and mutually exclusive relation can help us to generate reasonable event sequences, we compare the features generated by MLM model with different training data: (1) **Origin**: event sequences extracted from ROCStories Corpora. (2) **RandomWalk**: Randomly walk on the event graphs and sample training data. (3) **Graph-Plan**: Use our planning method to generate training data. Note that the input-event coherence score is excluded in the score function due to no input being given. We then use the event feature and adopt the state-of-the-art model DiffNet (Cui et al. 2019) for the story cloze test. For fair comparison, RandomWalk and GraphPlan sample the same amount of event chains in the dataset during training. Further details of the model could be found in the supplementary material.

Results of Story Cloze Test are presented in Table 6. It shows that RandomWalk and GraphPlan achieves better scores in both SCTv1.0 and v1.5, which prove that our event graphs and mutually exclusive events have positive effects on event planning.

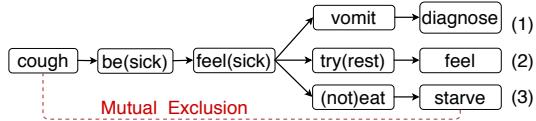
5.7 Other Story Generation Task

To further verify the effectiveness of our proposed method, we also conduct experiments on Visual Storytelling tasks. Due to the page limitation, we put it into the supplementary material. In the results, GraphPlan also shows improvement in terms of logicality.

6 Discussion

Controllable Generation As mentioned before, the stories can be easily controlled by modifying the events. Table 7 shows an example. Selecting different upcoming events for “feel(sick)” would change the following storyline.

Why GraphPlan Works? Our experiment shows that event graphs can produce more logical stories than planning with language models. Here we give an empirical explanation. Sequence-to-sequence models usually fail to capture long term relation and order information in the event sequence. The decoder is not guaranteed to take into account



-
- (1) *The man was coughing a lot. He was sick. He felt sick for days. He vomited on the couch. He was later diagnosed with the flu.*
-
- (2) *The man was coughing a lot. He was sick. He felt sick. He tried to rest for an hour. The man felt better!*
-
- (3) *The man was coughing a lot. He was sick. He felt sick. He couldn't eat anything. He starved himself.*
-

Table 7: Example of controllable generation. (1) and (2) extends different events after “feel sick” to achieve different endings and (3) shows logical inconsistency when generating with two mutually exclusive events

all previous events during decoding. At this point, our approach applying event-event coherence scores enforces the model to consider long term relations during planning. In addition, the order of events is captured from the gold cases that can be guaranteed in our event graphs. Moreover, mutually exclusive sets help us to decide whether two events can co-occur in one sequence no matter how distant two events are. Table 7 gives an example. “cough” and “starve” are considered as mutually exclusive events in our event graph. If we generate a story based on this event chain, the last sentence “he starved himself” seems not reasonable in this case.

The findings in this work also opens up new research questions: 1) Better event definition 2) Exposure Bias issue in the story generation module 3) Better topic modelling methods. These are left for our future work.

7 Conclusion

In this paper, we show that a graph-based event planning approach can indeed produce more natural event sequences compared to conventional language models. We propose to walk on automatically learned event graphs by performing beam search with a score function dedicated for event planning. Then the story is generated follows the planned events.

We evaluate our approach on event planning and open story generation with large scale human judgements. Results show that our proposed approach clearly outperforms the non-planning baseline and the sequence-to-sequence model based planning models. In human evaluation, the events and the stories generated by our proposal are believed to be more logical and coherent. An additional experiment on story cloze test further proves the advantages of event graphs and mutually exclusive sets.

References

- Ammanabrolu, P.; Tien, E.; Cheung, W.; Luo, Z.; Ma, W.; Martin, L. J.; and Riedl, M. O. 2019. Story Realization: Expanding Plot Events into Sentences. *arXiv preprint arXiv:1909.03480*.
- Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In

3rd International Conference on Learning Representations, ICLR 2015.

Bird, S.; Klein, E.; and Loper, E. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* ”O'Reilly Media, Inc.”

Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3(Jan): 993–1022.

Chambers, N.; and Jurafsky, D. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*, 789–797.

Chen, S.; Nelson, M. J.; Sullivan, A.; and Mateas, M. 2009. Evaluating the Authorial Leverage of Drama Management. In *AAAI Spring Symposium: Intelligent Narrative Technologies II*, 20–23.

Cui, Y.; Che, W.; Zhang, W.-N.; Liu, T.; Wang, S.; and Hu, G. 2019. Discriminative Sentence Modeling for Story Ending Prediction. *arXiv preprint arXiv:1912.09008*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Fan, A.; Lewis, M.; and Dauphin, Y. 2019. Strategies for structuring story generation. *arXiv preprint arXiv:1902.01109*.

Goldfarb-Tarrant, S.; Feng, H.; and Peng, N. 2019. Plan, Write, and Revise: an Interactive System for Open-Domain Story Generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, 89–97.

Granroth-Wilding, M.; and Clark, S. 2016. What Happens next? Event Prediction Using a Compositional Neural Network Model. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI'16*, 2727–2733. AAAI Press.

Holtzman, A.; Buys, J.; Du, L.; Forbes, M.; and Choi, Y. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Ippolito, D.; Grangier, D.; Callison-Burch, C.; and Eck, D. 2019. Unsupervised Hierarchical Story Infilling. In *Proceedings of the First Workshop on Narrative Understanding*, 37–43. Minneapolis, Minnesota: Association for Computational Linguistics. doi:10.18653/v1/W19-2405. URL <https://www.aclweb.org/anthology/W19-2405>.

Li, B.; Lee-Urban, S.; Johnston, G.; and Riedl, M. 2013. Story generation with crowdsourced plot graphs. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*.

Li, W.; Xu, J.; He, Y.; Yan, S.; Wu, Y.; and Sun, X. 2019. Coherent Comments Generation for Chinese Articles with a Graph-to-Sequence Model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 4843–4852. Florence, Italy: Association for Computational Linguistics. doi:10.18653/v1/P19-1479. URL <https://www.aclweb.org/anthology/P19-1479>.

Martin, L. J.; Ammanabrolu, P.; Wang, X.; Hancock, W.; Singh, S.; Harrison, B.; and Riedl, M. O. 2018. Event representations for automated story generation with deep neural nets. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Mateas, M.; and Sengers, P. 2003. *Narrative intelligence*. J. Benjamins Pub.

McIntyre, N.; and Lapata, M. 2010. Plot induction and evolutionary search for story generation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 1562–1572. Association for Computational Linguistics.

Meehan, J. R. 1976. The metanovel: writing stories by computer. Technical report, YALE UNIV NEW HAVEN CONN DEPT OF COMPUTER SCIENCE.

Mostafazadeh, N.; Roth, M.; Louis, A.; Chambers, N.; and Allen, J. 2017. Lsdsem 2017 shared task: The story cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, 46–51.

Peng, H.; and Roth, D. 2016. Two Discourse Driven Language Models for Semantics. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 290–300. Berlin, Germany: Association for Computational Linguistics. doi:10.18653/v1/P16-1028. URL <https://www.aclweb.org/anthology/P16-1028>.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language models are unsupervised multitask learners. *OpenAI Blog* 1(8): 9.

Regneri, M.; Koller, A.; and Pinkal, M. 2010. Learning script knowledge with web experiments. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 979–988.

Reiter, E.; and Dale, R. 1997. Building applied natural language generation systems. *Natural Language Engineering* 3(1): 57–87.

Riedl, M. O. 2010. Story planning: Creativity through exploration, retrieval, and analogical transformation. *Minds and Machines* 20(4): 589–614.

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

Tambwekar, P.; Dhuliawala, M.; Martin, L. J.; Mehta, A.; Harrison, B.; and Riedl, M. O. 2018. Controllable Neural Story Plot Generation via Reinforcement Learning. *arXiv preprint arXiv:1809.10736*.

Tambwekar, P.; Dhuliawala, M.; Martin, L. J.; Mehta, A.; Harrison, B.; and Riedl, M. O. 2019. Controllable neural story plot generation via reward shaping. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 5982–5988. AAAI Press.

Tan, B.; Yang, Z.; Al-Shedivat, M.; Xing, E. P.; and Hu, Z. 2020. Progressive Generation of Long Text. *arXiv preprint arXiv:2006.15720*.

Weyhrauch, P. 1997. *Guiding interactive fiction*. Ph.D. thesis, Ph. D. Dissertation, Carnegie Mellon University.

Xu, J.; Ren, X.; Zhang, Y.; Zeng, Q.; Cai, X.; and Sun, X. 2018. A Skeleton-Based Model for Promoting Coherence Among Sentences in Narrative Story Generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 4306–4315.

Yao, L.; Peng, N.; Weischedel, R.; Knight, K.; Zhao, D.; and Yan, R. 2019. Plan-and-write: Towards better automatic storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 7378–7385.