GraphPlan: Story Generation by Planning with Event Graph

Hong Chen[1,3], **Raphael Shu**[1], **Hiroya Takamura**[2,3], **Hideki Nakayama**[1,3]
The University of Tokyo[1], Tokyo Institute of Technology[2]
National Institute of Advanced Industrial Science and Technology, Japan[3]
{chen, nakayama}@nlab.ci.i.u-tokyo.ac.jp
shu@deeplearn.org, takamura.hiroya@aist.go.jp

Abstract

Story generation is a task that aims to automatically generate a meaningful story. This task is challenging because it requires high-level understanding of the semantic meaning of sentences and causality of story events. Naive sequence-to-sequence models generally fail to acquire such knowledge, as it is difficult to guarantee logical correctness in a text generation model without strategic planning. In this study, we focus on planning a sequence of events assisted by event graphs and use the events to guide the generator. Rather than using a sequence-to-sequence model to output a sequence, 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 incorporate human participation, both in event planning and story generation. Based on the large-scale human annotation results, our proposed approach has been shown to provide more logically correct event sequences and stories compared with previous approaches.

1 Introduction

Narrative intelligence (Mateas and Sengers, 2003) is a form of humanistic artificial intelligence that requires the system to organize, comprehend, and reason about narratives, and then, 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. In addition to leaving the model to output random sequences, the model is usually given a specific topic (e.g., title or prompt) or visual information (e.g., image or video). One straightforward 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 the training data, it has two serious drawbacks when applied to generating stories: 1) A conditional language model (i.e., the decoder) tends to assign high probabilities to generic, repetitive words, especially when beam search is applied in the decoding phase (Holtzman et al., 2019); and 2) sequence-to-sequence models often fail to produce logically correct stories.

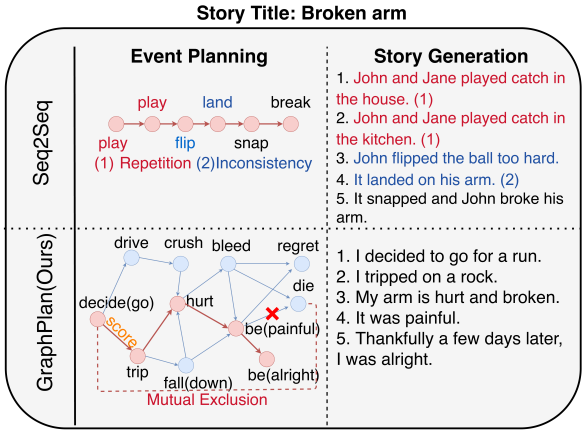


Figure 1: Comparison between sequence-to-sequence model and GraphPlan (ours). Two problems occur in the sequence-to-sequence model when generating events: repetition and logical inconsistency. Repeated words (e.g., play) in the storyline result in repeated sentences in the generated stories. In addition, 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, and applies beam search on the event graph based on a well-designed score function. The mutually exclusive set further ensures global logical consistency for the planned events.

Recently, there has been significant interest in decomposing story generation into two phases: Planning and generation (Yao et al., 2019; Goldfarb-Tarrant et al., 2019; Xu et al., 2018; Fan et al., 2019). Planning (Meehan, 1976; Riedl and Young, 2010) creates a high-level abstraction or a blueprint

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 377–386,
Aberdeen, Scotland, UK, 20-24 September 2021.

that encourages 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 several 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): tuple of subject, verb, object and modifier or "wildcard" (e.g., *(PERSON0, correspond-36.1, empty, PERSON1)*) (Martin et al., 2018; Ammanabrolu et al., 2020) or reconstructed verb phrase (e.g., *decide(go)*) (Peng and Roth, 2016). In this paper, we follow Peng and Roth (2016) to represent an event with verb phrases.

Existing approaches (Goldfarb-Tarrant et al., 2019; Martin et al., 2018; Ammanabrolu et al., 2020) regard event generation as an abstracted case of story generation. In other words, they treat each event as one token and use a sequence-to-sequence model to plan the events. Our preliminary experiments show that repetition and logical inconsistency problems occur in the event sequence; further, the same problems occur in the generated stories. Figure 1 shows an example using sequence-to-sequence event planning. Both events and stories are repeated and illogical.

In this study, instead of a sequence-to-sequence model for event planning, we propose a planning method, **GraphPlan**. To plan the event, GraphPlan walks on a topic-specific event graph with a beam search. Event graphs have been adopted for story generation even before the emergence of neural-based models (Weyhrauch, 1997; Chen et al., 2009; Regneri et al., 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. We can walk on a learned event graph and produce a reasonable event sequence. We follow the graph setting in Li et al. (2013), wherein each graph is composed of event nodes, functions 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 subsequently, retrieve a related event graph. We then plan the events by running a beam search with a score function that considers event-event coherence and input-event coherence into account. Finally, a *story generation* module transforms the planned event sequence into a readable story. Figure 2 depicts the entire pipeline of our proposed approach.

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

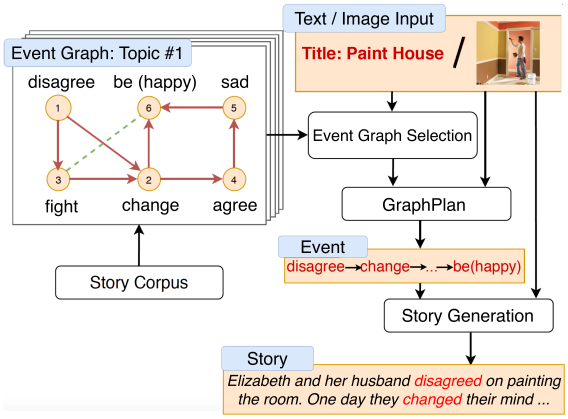


Figure 2: Overview of our approach. In the preprocessing step, we cluster the stories into K topics and build an event graph for each topic. In the planning step, an event graph selection module selects an event graph based on the input. Then, a related event graph is retrieved. The event planning model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

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

2 Related Work

Planning for Story Generation Several approaches have been explored to plan the skeleton of a story before its 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 have explicitly plan a sequence of events (Martin et al., 2018; Ammanabrolu et al., 2020; Tambwekar et al., 2019; Fan et al., 2018; Rashkin et al., 2020; Ammanabrolu et al., 2021), keywords (Yao et al., 2019; Ippolito et al., 2019; Goldfarb-Tarrant et al., 2020) or actions (Fan et al., 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 results 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 An event graph is a variant of a plot graph whose nodes represent events. Previous research has made progress on generating stories from plot graphs (Weyhrauch, 1997; Chen et al., 2009; Regneri et al., 2010; McIntyre and Lapata, 2010; Li et al., 2019). Li et al. (2013) proposed a plot graph on story generation tasks, which is relevant to our work. They crowdsourced 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 that considers 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 that is abstract, simple, and comprehensible for humans. From our observation, this representation does not have a severe sparseness problem. In statistics, each event in the graph can link to over three next possible events on average. Note that our work does not investigate event representation; instead, we focus on planning a more logical event sequence. Specifically, as a preprocessing step, we parse all sentences with semantic role labeling [1] and extract the verb phrases. If an extracted verb has an argument with the 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 using NLTK (Bird et al., 2009).

Topic Modelling Generally, a story dataset contains various topics, ranging from animals to health to robbery. Here, we use Latent Dirichlet Allocation (LDA) (Blei et al., 2003) to infer the topics in the corpus. Considering that the relation between events drastically changes according to the topic, in this study, we build an independent event graph for each topic. Formally, we denote $e_1^k, \ldots, e_t^k$ 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 events from a corpus for each topic, we need to determine 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 is presented 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 thus, cannot be placed in the same story. These mutually exclusive relations are considered as exceptions, which are difficult to represent along with the event graph. We create a held-out set consisting of mutually exclusive event pairs for each graph.

To identify these mutually exclusive events from the constructed graphs, we prepare an event-event

[1] https://demo.allennlp.org/semantic-role-labeling/semantic-role-labeling

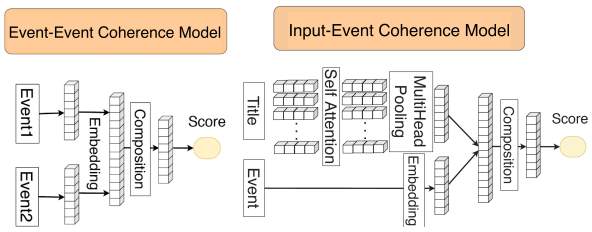


Figure 3: Coherence models were used in this study. The event-event coherence model outputs a coherence score for two events. The input-event coherence model takes a title and an event as input. Both coherence models finally produce a score within 0 to 1. These coherence scores determine the next event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

coherence model to detect the coherence score between two events. Consequently, 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 samples are randomly sampled from the events in different stories. Let (e_i, e_j) denote a positive pair of events, and $\tilde{e}_j$ denotes 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 as mutually exclusive if the coherence score falls below a certain threshold τ.

On average, after considering the mutually exclusive sets, each event graph can still plan over one million different possible event sequences. Please refer to the supplementary materials for more statistical details on 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 for our work.

4 GraphPlan: Planned Story Generation using Event Graph

In this section, we describe our approach for planned story generation. We separated the entire 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 text following the planned events.

4.1 GraphPlan

Each topic has had 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 comprises two steps. 1) Selecting an event graph for the input (i.e. title or image); and 2) generating an event sequence by walking on the graph.

Event Graph Selection First, we identify the topic of inputs to retrieve the corresponding event graph. Depending on the 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, 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, 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:

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

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

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 the score function sums the event-event coherence score of candidate event e_t to each partially generated event e_i and gives

more weight to recent events. λ denotes the decay rate. Then, a decayed average is applied over the score. The model used in producing the event-event coherence is the same model that is used to detect 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; refer to Figure 3 for details about parameterization. For the task of open story generation, the input-event coherence model is implemented via compositional neural networks, where input x in Equation (3) is the title. As common practice for beam search, we set a budget for 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.

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 module can be any type of model. Recently, large pre-trained language models show remarkable capacity for generating knowledgeable and informative sentences. Utilizing 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 events and another special token "<SEP>" is placed in every interval of the events. "<|endofinput|>" token is added at the end of the input. In addition, we 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), the exposure bias problem occurs when plan-write strategy is applied. To mitigate this problem, we alternatively add two types of noises into the inputs: 1) Mask 20% events with a "[MASK]" token; and 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 stories related 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 and story quality. First, 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 stories with titles that we use as the input and a five-sentence story that we use as the target. We chose this dataset as a testbed since the sentences included inside are simple, making it easier to accurately capture the events. We split these stories into 8:1:1 for training, validation 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.

5.3 Baseline

S2S Following Yao et al. (2019), we use a sequence-to-sequence model (Bahdanau et al., 2015), which straightforwardly generates events by the titles.
S2S(R) To build a more competitive baseline, we adopt reward shaping in the sequence-to-sequence model. As in (Tambwekar et al.), we apply a policy gradient on

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

where e denotes the 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 $R(e_i)$, which is propor-

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. Both sequence-to-sequence models achieve low diversity, while Graph-Plan can achieve high diversity.

tional to $r_1(e_i)$ and $r_2(e_i)$. In brief, $r_1(e_i)$ increases 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 forces the model to focus on the event that has a high coherence score with 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 aim to show the importance of using coherence models by comparing it with this method.

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

5.4 Event Quality

We plan the events on 1000 randomly selected test data with different baselines models and our proposed model. We first tested the diversity of the generated sequences and calculate Distinct-1 and Distinct-2 scores to measure the diversity for all generated events. Table 1 shows that the sequence-to-sequence model suffers from producing repeated unigram and bigram events. Graph plan produces 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 in (Tambwekar et al., 2019), we use the verb phrase, which is more easily understood by humans. Thus, we request the annotators to compare the event sequences using two criteria: Relevance and logicality. Table 2 shows the human evaluation results. These results show 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, they tend to output the events that occur with high frequency in the corpus, such as "be"(there is sth.) and "go" (sb. go somewhere). This is common for a model

Choices(%)	GP vs S2S		GP vs S2S(R)		GP vs GR	
	GP	S2S	GP	S2S(R)	GP	GR
Relevance	**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 of event planning. Cohen's Kappa coefficient (κ) for all annotations is in moderate agreement (0.4-0.6). Sign tests show that 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	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.

trained under the framework of the maximum likelihood estimation method. Although reward shaping (S2S(R)) helps with this problem substantially, it is still not eliminated. Without the limitation of the coherence score, GR walks on graphs randomly to produce a sequence. As the graph is not small, achieving a good event sequence is extremely challenging. Our proposed GP produces more logical and diverse events, using which humans can easily tell a story based on these given events.

5.5 Open Story Generation

The goal of event planning is to generate more related and logical coherent stories. Human evaluation of the event sequence is subjective and tricky since the event is highly abstract. To prove that better event planning improves 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 the story generation module (i.e. GPT2 and RNN) to transform these planned events into the full stories. We compare the following methods:

GPT2. This is a large scale language model that has shown remarkable performance in generating stories in recent research. In this method, we directly input the title to GPT2 and generate full stories.

***+GPT2**. We associate the event planning meth-

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
Relevance	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 of open story generation. (+GPT2) are omitted for all methods, except for GP+RNN. We calculate Cohen's Kappa coefficient (κ). All the results are in moderate agreement (0.4-0.6). We conduct sign tests to show the significant difference. (*) and (**) denotes p-value is < 0.05 and < 0.01 respectively. The results show that our method achieves significant improvement on logicality, thereby getting better overall performance.

	Title: Fire next door
GPT-2	*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)*
S2S	make → think → go → see → get → apologize
	*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**.*
S2S(R)	try → tell → leave → throw → take
	*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.*
GR	quit → be → look(buy) → find → go
	*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.*
GP+RNN	catch(on) → come → go(off) → evacuate → extinguish
	*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.*
GP	catch(on) → come → go(off) → evacuate → extinguish
	*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.*
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. Red words represent the events.

ods with GPT2, which is used in the story generation module. Thus, we compare 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 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 conducted 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 the human evaluation are listed in the supplementary materials. We randomly sampled 300 titles from the testing set and generated stories using each method. Pairwise comparison was conducted for each criterion and each sample was assigned to two different workers to avoid randomness or personal bias. Table 4 shows that our approach performs better with respect to logicality and overall. In particular, our method greatly outperforms other planning methods in the logicality measure, which suggests that our planned events are logically sound. We believe that the following two factors are the primary reasons for improved logic: 1) Each event graph is built from the corpus; thus, walking on the graph retains 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 examples of stories generated using these methods. We show both the planned events and 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, leading to a low logicality score in the generated sentences. Because there is no restriction on the event selection in GR, the event produced could be irrelevant to the title and even mutually contradictory. Using our proposed method, GP can plan a reasonable set of events, and thus, generate the most logical story.

5.6 Story Cloze Test (SCT)

To better validate the effectiveness of our event graphs and the mutually exclusive relation between events, we conducted SCT. This task aims to select the correct ending sentence from two candidates. We incorporated the event features generated by

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

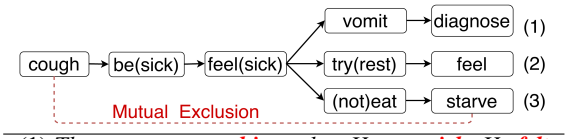
Table 6: Results on story cloze test. From the results, events planned by our event graphs and mutually exclusive sets have positive effects on this task.

different methods into the SCT. The accuracy of SCT reflects the quality of the event features. The event feature is learned by a mask language model (MLM) (i.e., the BERT model with fewer parameters) (Devlin et al., 2019). If the training event sequence is more logical and reasonable, the feature learned by MLM would better fit the SCT. To prove that our event graph and mutually exclusive relation can help us to generate reasonable event sequences, we compared the features generated by the MLM model with different training data: (1) **Origin**: Event sequences extracted from ROCStories Corpora. (2) **RandomWalk**: Random walk on the event graphs and sample training data. (3) **GraphPlan**: Using our planning method to generate training data. Note that the input-event coherence score is excluded in the score function because no input being given. We then use the event feature and adopt the state-of-the-art model, DiffNet (Cui et al., 2020) for SCT. For fair comparison, RandomWalk and GraphPlan included the same number of event chains in the dataset during training. Further details of the model can be found in the supplementary material.

Table 6 presents the results of SCT. This shows that RandomWalk and GraphPlan achieve better scores in both SCTv1.0 and v1.5, proving that our event graphs and mutually exclusive events have positive effects on event planning.

6 Discussion

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)" will change the following storyline. Moreover, our experiment shows that event graphs can produce more logical stories than planning via language models. Here, we give an empirical explanation. Sequence-to-sequence models usually fail to capture long-term relations and order information in the event sequence. The decoder is not guaranteed to account for all previous events during



(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

decoding. At this point, our approach applies event-event coherence scores, which forces the model to consider long-term relations during planning. In addition, the order of events is captured from the gold cases, which can be guaranteed in our event graphs. Moreover, mutually exclusive sets help us to determine whether two events can co-occur in one sequence regardless of the distance between two events. Table 7 provides an example. Note that "cough" and "starve" are considered mutually exclusive events in our event graph. If we generate a story based on this event chain, the last sentence "he starved himself" is unreasonable in this case.

7 Conclusion

In this study, we demonstrate that a graph-based event planning approach can indeed produce more natural event sequences compared with conventional language models. We propose to walk on automatically learned event graphs by performing beam search using 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. The 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 proposed method 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.

8 Acknowledgements

We thank the anonymous reviewers for the useful comments. This work was supported by JSPS KAKENHI Grant Numbers JP19H04166 and based on results obtained from a project JPNP20006, commissioned by the New Energy and Industrial Technology Development Organization (NEDO). For experiments, computational resource of AI Bridging Cloud Infrastructure (ABCI) provided by National Institute of Advanced Industrial Science and Technology (AIST) was used.

References

Prithviraj Ammanabrolu, Wesley Cheung, William Broniec, and Mark O Riedl. 2021. Automated storytelling via causal, commonsense plot ordering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 5859–5867.

Prithviraj Ammanabrolu, Ethan Tien, Wesley Cheung, Zhaochen Luo, William Ma, Lara J Martin, and Mark O Riedl. 2020. Story realization: Expanding plot events into sentences. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7375–7382.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

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

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*, pages 789–797.

Sherol Chen, Mark J Nelson, Anne Sullivan, and Michael Mateas. 2009. Evaluating the authorial leverage of drama management. In *AAAI Spring Symposium: Intelligent Narrative Technologies II*, pages 20–23.

Yiming Cui, Wanxiang Che, Wei-Nan Zhang, Ting Liu, Shijin Wang, and Guoping Hu. 2020. Discriminative sentence modeling for story ending prediction. In *AAAI*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Angela Fan, Mike Lewis, and Yann Dauphin. 2019. Strategies for structuring story generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2650–2660.

Seraphina Goldfarb-Tarrant, Tuhin Chakrabarty, Ralph Weischedel, and Nanyun Peng. 2020. Content planning for neural story generation with aristotelian rescoring. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4319–4338.

Seraphina Goldfarb-Tarrant, Haining Feng, and Nanyun Peng. 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)*, pages 89–97.

Mark Granroth-Wilding and Stephen Clark. 2016. What happens next? event prediction using a compositional neural network model. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2727–2733. AAAI Press.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Daphne Ippolito, David Grangier, Chris Callison-Burch, and Douglas Eck. 2019. Unsupervised hierarchical story infilling. In *Proceedings of the First Workshop on Narrative Understanding*, pages 37–43, Minneapolis, Minnesota. Association for Computational Linguistics.

Boyang Li, Stephen Lee-Urban, George Johnston, and Mark Riedl. 2013. Story generation with crowdsourced plot graphs. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*.

Wei Li, Jingjing Xu, Yancheng He, ShengLi Yan, Yunfang Wu, and Xu Sun. 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*, pages 4843–4852, Florence, Italy. Association for Computational Linguistics.

Lara J Martin, Prithviraj Ammanabrolu, Xinyu Wang, William Hancock, Shruti Singh, Brent Harrison, and Mark O Riedl. 2018. Event representations for automated story generation with deep neural nets. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Michael Mateas and Phoebe Sengers. 2003. *Narrative intelligence*. J. Benjamins Pub.

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

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

Nasrin Mostafazadeh, Michael Roth, Annie Louis, Nathanael Chambers, and James Allen. 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*, pages 46–51.

Haoruo Peng and Dan Roth. 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)*, pages 290–300, Berlin, Germany. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Hannah Rashkin, Asli Celikyilmaz, Yejin Choi, and Jianfeng Gao. 2020. PlotMachines: Outline-conditioned generation with dynamic plot state tracking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4274–4295, Online. Association for Computational Linguistics.

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

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

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

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

Pradyumna Tambwekar, Murtaza Dhuliawala, Lara J Martin, Animesh Mehta, Brent Harrison, and Mark O Riedl. Controllable neural story plot generation via reward shaping.

Pradyumna Tambwekar, Murtaza Dhuliawala, Lara J Martin, Animesh Mehta, Brent Harrison, and Mark O Riedl. 2019. Controllable neural story plot generation via reward shaping. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 5982–5988. AAAI Press.

Bowen Tan, Zichao Yang, Maruan AI-Shedivat, Eric P Xing, and Zhiting Hu. 2020. Progressive generation of long text. *arXiv preprint arXiv:2006.15720*.

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

Jingjing Xu, Xuancheng Ren, Yi Zhang, Qi Zeng, Xiaoyan Cai, and Xu Sun. 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*, pages 4306–4315.

Lili Yao, Nanyun Peng, Ralph Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-and-write: Towards better automatic storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7378–7385.