NGEP: A Graph-based Event Planning Framework for Story Generation

Chen Tang[1], Zhihao Zhang[2], Tyler Loakman[3], Chenghua Lin[3]* and Frank Guerin[1]
[1]Department of Computer Science, The University of Surrey, UK
[2]School of Economics and Management, Beihang University, Beijing, China
[3]Department of Computer Science, The University of Sheffield, UK
{chen.tang,f.guerin}@surrey.ac.uk
zhhzhang@buaa.edu.cn
{tcloakman1,c.lin}@sheffield.ac.uk

Abstract

To improve the performance of long text generation, recent studies have leveraged automatically planned event structures (i.e. storylines) to guide story generation. Such prior works mostly employ end-to-end neural generation models to predict event sequences for a story. However, such generation models struggle to guarantee the narrative coherence of separate events due to the hallucination problem, and additionally the generated event sequences are often hard to control due to the end-to-end nature of the models. To address these challenges, we propose NGEP, an novel event planning framework which generates an event sequence by performing inference on an automatically constructed event graph and enhances generalisation ability through a neural event advisor. We conduct a range of experiments on multiple criteria, and the results demonstrate that our graph-based neural framework outperforms the state-of-the-art (SOTA) event planning approaches, considering both the performance of event sequence generation and the effectiveness on the downstream task of story generation.

1 Introduction

Current neural generation models struggle to generate long stories as it is difficult to guarantee the logical coherence of generated sentences when conditioning only on a limited size input (e.g. leading context or title). Therefore, current story generation frameworks are usually split into two stages, planning and writing, using an automatically planned storyline (aka. event sequence) (Alhussain and Azmi, 2021; Tang et al., 2022) as the intermediate between planning and writing.

In order to plan an event sequence, prior works (Martin et al., 2018; Yao et al., 2019; Chen et al., 2021; Alhussain and Azmi, 2021; Wang et al., 2020) mostly focus on leveraging end-to-end neural generation models, such as BART (Lewis et al., 2020), to predict events. However, whilst some efforts (Goldfarb-Tarrant et al., 2020; Ahn et al., 2016) have been made to improve neural event planning (e.g., Goldfarb-Tarrant et al. (2020) use rescoring models to guide the planning process), event planning based on neural generation models still tends to suffer from common limitations: (i) The selection of individual events in the sequence is hard to control (because of the end-to-end generation) (Chen et al., 2021); and (ii) Due to the hallucination problem (Rohrbach et al., 2018; Elder et al., 2020; Cheng et al., 2021; Tang et al., 2022) each predicted event is not guaranteed to be complete and accurate.

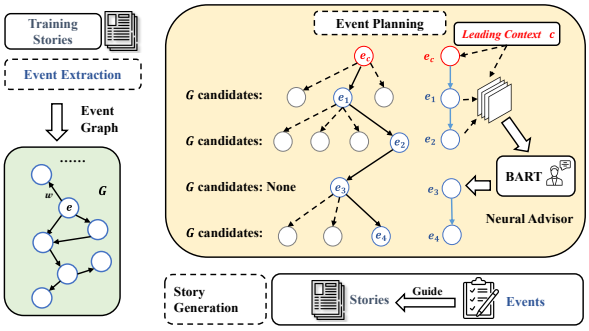

Figure 1: The overview of our proposed NGEP model. The event graph G is automatically constructed from the training set, and the potential event candidates are generated according to the conditional probability distribution modelled on G when event planning. If there are no proper candidates for the next event, we leverage a BART-based neural advisor to predict the best choice.

In this study, we propose **NGEP**, a novel **N**eural **G**raph-based **E**vent **P**lanning framework to predict event sequences for story generation. An overview of the proposed framework is illustrated in Figure 1. Firstly, events are extracted from the training set in order to construct an event graph which records the events and their neighbour relations. This graph can then be used at test-time to predict events from a leading context. The conditional probability distribution is modelled by a coherence score calcu-

*Corresponding author.

Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pages 186–193
November 20–23, 2022.

lated with the degrees of event nodes and the concurrency of predicted events. When an event graph is unable to generate event candidates, i.e. no edges point to another event, a BART-based neural advisor is introduced to predict the next event from the graph. The neural advisor is trained to model the conditional probability between event nodes and the context, including the input and previously predicted events, so that it can predict the next individual event rather than the entire sequence, thus enhancing controllability. Finally, the predicted event sequence is sent to a downstream model for story generation.

To the best of our knowledge, this is the first attempt to employ an unsupervised graph-based inference approach with a neural advisor as the event planning framework. A range of experiments are conducted to evaluate the performance of our approach, both on the quality of event sequences and their efficacy in aiding story generation. The results demonstrate that our model significantly outperforms all competitive baselines.[1]

2 Methodology

The story generation task is formulated as follows: The given input is a sentence acting as the leading context $C = \{c_1, c_2, ..., c_n\}$ where c_i denotes the i-th token of the leading context, and the output is a multi-sentence story $S = \{s_1^1, s_2^1, ..., s_1^2 ..., s_n^m\}$, where s_j^i denotes the j-th token of i-th sentence in a story. The task requires the prediction of an event sequence[2] $E = \{e_1, e_2, ..., e_m\}$ as a intermediate input, which is generated according to the leading context C and used to generate a story S. e_i denotes the i-th event representing the i-th sentence in a story, and each event may have multiple tokens.

2.1 Event Graph Construction

The representation of an event is defined as a verb phrase that describes the main event within a sentence. We employ *spaCy*[3] to parse dependencies between words in a given sentence, and then extract all key roles to compose an event. Neighboring events are considered to have directed relations r (previous/next event), so that each story may contain several triplets $\{e_{\text{head}}, r, e_{\text{tail}}\}$. The set of all triplets in the training set is the event graph G. The sum of repeated triplets of an event in the training set is recorded as weighted degrees d in G for calculations of the conditional probability between events. Due to space constraints, the details of the event schema and extraction framework are described in the Appendix (A.1 and A.2, respectively).

2.2 Graph-based Event Planning

Due to there being no single unique storyline for a given topic, we argue that the planned event sequences for open-domain story generation should instead focus on the intrinsic relatedness between events and their relevance to the leading context. Therefore, we reference the framework of Bamman and Smith (2014) and propose an unsupervised graph-based approach to model the conditional probability distribution between events in the event graph G. The event contained within the leading context denoted as e_c is set to be the start of the event planning process. Let $P(e_i'|E^c_{e_{t<i}}, G)$ denote the conditional probability of candidates for the i-th event e_i, and $E^c_{e_{t<i}} = \{e_c, e_1, ...e_{i-1}\}$ denote the input of prior events for the prediction of e_i. $P(e_i'|E^c_{e_{t<i}}, G)$ is calculated as follows:

$$P(e_i'|E^c_{e_{t<i}}, G) = \frac{f_s(r(e_{i-1}, e_i'))}{\sum_{r(e_{i-1},*)\in G} f_s(r)} \quad (1)$$

$$f_s(r(e_{i-1}, e_i')) = \omega(e_{i-1}, e_i') d_{e_i'} \times \gamma_(e_i'|E^c_{e_{t<i}}) \quad (2)$$

$$\gamma_(e_i'|E^c_{e_{t<i}}) = \frac{\left|rept_m - c^-(e_i', E^c_{e_{t<i}})\right|}{rept_m \times d^{in}_{*e_i'}} \quad (3)$$

$$e_i \overset{sampling}{\Longleftarrow} P(e_i'|E^c_{e_{t<i}}, G)) \quad (4)$$

where $\gamma_(e_i'|E^c_{e_{t<i}})$ denotes the repetition penalty of a candidate e_i' ranging from 0 to 1, and $rept_m$ denotes the maximum number of repetitions permitted in $E^c_{e_{t<i}}$. We penalise candidates with its weighted in-degree $d^{in}_{*e_i'}$, as this means it has a relatively weak relationship to e_{i-1}. $c^-(e_i', E^c_{e_{t<i}})$ counts the occurrences of e_i' observed in $E^c_{e_{t<i}}$.

$f_s(r(e_{i-1}, e_i'))$ is the event score function which evaluates the probability of event e_i' through the calculation of the weight of edge $\omega(e_{i-1}, e_i')$ (as the graph is isomorphic, we set it to 1 here) and the degrees of the event node $d_{e_i'}$. Furthermore, $r(e_{head}, e_{tail})$ denotes the directed edge from the head event pointing to the tail event, with $*$ acting

[1] Our code for reproduction is available at https://github.com/tangg555/NGEP-eventplan.

[2] We combine events with special tokens, e.g., "<*s*> needed get <*sep*> ... <*e*>", where "<*s*>","<*sep*>", "<*e*>" denote the start, separation, and end of planning, respectively.

[3] https://spacy.io/

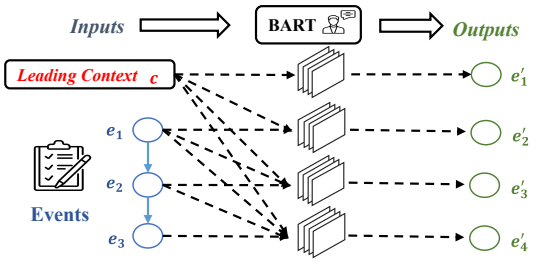

Figure 2: Illustration of the neural advisor.

as the wildcard character representing any available event. $P(e'_i|E^c_{e_{t<i}}, G)$ is calculated using the event score function and the repetition penalty. Finally, we select the candidate e'_i by sampling candidates according to the probability distribution P.

2.3 Neural Advisor

Event graph inference may not be possible for all instances in the test set if the extracted event from a leading context has not been seen at graph construction time. Consequently, if the event graph is unable to generate any candidates for the next event we need another module to analyse the given information and predict the most probable candidate to compose the storyline. Therefore, as Figure 2 shows, we train a generation model, BART, to "advise" on selecting the next event as below:

$$E_{e_{t<i}}\{e_1, ..., e_{i-1}\} \quad \text{s.t.} e_t \in G \tag{5}$$

$$F_i = Encoder([C; E_{e_{t<i}}]) \tag{6}$$

$$e'_i \overset{predict}{\Longleftarrow} Decoder(F_i) \tag{7}$$

where $E_{e_{t<i}}$ denotes the prior event sequences before time step i. When training, we force BART to learn the relations between reference events, and then find the closest event candidate e'_i via the Jaccard similarity index in G to be the next event e_i.

2.4 Overall Event Planning Process

We combine the graph-based event planning with the neural advisor (denoted as $advise(*)$) to predict event sequences (illustrated in algorithm 1). The training objective of neural advisor is same to the vanilla BART, and the graph-based event planning process is unsupervised.

Algorithm 1: Predict Event Sequence E

Input: A leading context C and the event graph G, the minimal planning size of events $l_{\min}$ and the maximal $l_{\max}$

Output: Event Sequence E for C

Initialize $E \leftarrow []$;
extract e_c from C
if $e_c \notin G$ **then**
  reselect $e_c \leftarrow e'_c \in G$ where e'_c is equal $e_c.verb$, otherwise $e_c \leftarrow advise(e_c)$
$e_{\text{pre}} \leftarrow e_c$
while $|E| < l_{min}$ or $|E| > l_{max}$ **do**
  Let E' denote the set of candidates e'_{next}
  if $E' = \varnothing$ **then**
    $e_{\text{next}} \leftarrow advise(e_{\text{next}})$
  else
    Get $\gamma(e'_{\text{next}}|E^c_{e_{t<\text{next}}})$ for E'
    Get $P(e'_i|E^c_{e_{t<i}}, G)$ for E'
    Sample e_{next} according to P
  Append e_{next} to E

3 Experiment

3.1 Experiment Setup

Datasets We conduct our experiments on ROCStories (Mostafazadeh et al., 2016), following the work of Guan et al. (2021) to preprocess and split the data. The total number of stories in the Train/Dev/Test sets is 88344/4908/4909.

Training Details and Parameters Experiments were performed on an RTX A5000 GPU, and the random seed was fixed to 42 to facilitate reproduction. We implement the PyTorch Lightning[4] framework to set up training processes. The training parameters are as follows: *batch size* is set to 64; *learning rate* is 1e−4; *max source length* is set to 1024; the optimiser uses $Adam$ (Kingma and Ba, 2014), and the ϵ of $Adam$ is set to 1e−8. The whole training process runs for 5 *epochs*, but the results only consider the checkpoint with the best performance (lowest loss).

Metrics	R-1↑	R-2↑	R-L↑	B-1↑	B-2↑	D-1↑	D-2↑
Seq2Seq	54.33	29.10	53.05	0.391	0.089	0.051	0.277
BART	56.36	30.35	54.68	0.398	0.095	0.060	0.298
GPT-2	44.78	20.71	42.80	0.217	0.052	0.055	**0.318**
EventAdvisor	**59.85**	**32.43**	**57.74**	**0.436**	**0.110**	0.050	0.257
NGEP	59.30	31.96	57.54	0.429	0.099	**0.072**	0.311
Golden	N/A	N/A	N/A	N/A	N/A	0.072	0.315

Table 1: Automatic evaluation on event sequences. ↑ / ↓ means the higher/lower the metric, the better. The best performing model is highlighted in **bold**, and the second best is underlined.

[4] https://www.pytorchlightning.ai/

	Seq2Seq$_{story}$				BART$_{story}$				HINT$_{story}$				T-5$_{story}$			
	IR-A↓	D-2↑	D-3↑	D-4↑	IR-A↓	D-2↑	D-3↑	D-4↑	IR-A↓	D-2↑	D-3↑	D-4↑	IR-A↓	D-2↑	D-3↑	D-4↑
w/o events	**1.16**	0.233	0.554	0.777	1.88	0.243	0.567	0.789	1.81	0.188	0.494	0.740	1.68	0.216	0.498	0.719
Seq2Seq	1.27	0.227	0.546	0.773	1.40	0.247	0.576	0.799	1.43	0.185	0.490	0.738	1.54	0.213	0.497	0.719
BART	1.33	0.230	0.547	0.769	1.74	0.250	0.575	0.795	1.76	0.188	0.490	0.732	1.93	0.218	0.498	0.719
GPT-2	1.25	0.222	0.544	0.776	1.98	0.235	0.565	0.791	1.87	0.174	0.472	0.720	2.32	0.209	0.493	0.718
EventAdvisor	1.32	0.234	0.555	0.778	1.75	0.244	0.564	0.781	1.80	0.183	0.478	0.718	1.84	0.211	0.490	0.712
NGEP	**1.16**	**0.235**	**0.558**	**0.779**	**1.31**	**0.272**	**0.601**	**0.811**	**1.25**	**0.244**	**0.507**	**0.742**	**1.29**	**0.231**	**0.517**	**0.738**

Table 2: Automatic evaluation with unreferenced metrics on generated stories. The row labels stand for different event planning methods, and the column labels are SOTA models for story generation.

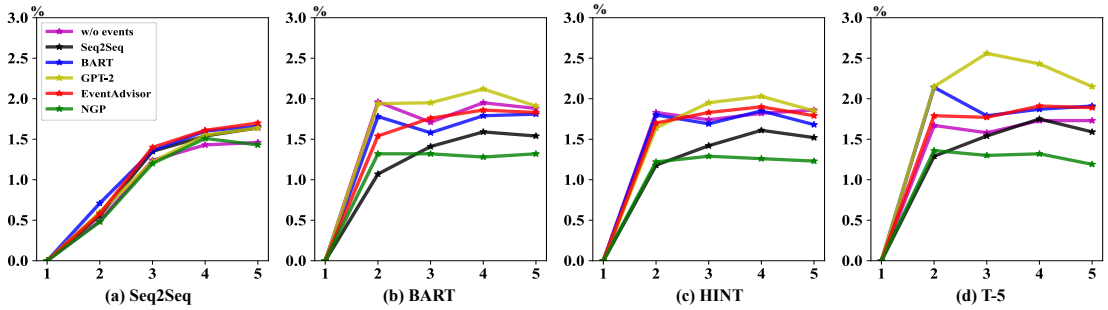

Figure 3: Intra-story repetitions (the lower the better) for each sentence in a story. We show the performance of different event planning approaches work different story generation models.

Baselines Several SOTA generation models for event planning and story generation (or long text generation) are selected as baselines.[5] (i) **Neural Event Planning:** Seq2Seq (Yao et al., 2019), BART (Goldfarb-Tarrant et al., 2020), and GPT-2 (Chen et al., 2021); (ii) **Story Generation** Seq2Seq (Yao et al., 2019), BART (Goldfarb-Tarrant et al., 2020), HINT (Guan et al., 2021), and T-5 (Raffel et al., 2020), in line with previous work in the area.

3.2 Evaluation Metrics

We adopt a range of automatic metrics including **ROUGE-n (R-n)** (Lin, 2004) and **BLEU-n (B-n)** (Papineni et al., 2002) as referenced metrics to compare to human-written event plans, and **Distinction-n (D-n)** (Li et al., 2016), **Intra-story Repetition** (Yao et al., 2019), and **Intra-story Repetition Aggregate Score (IR-A)** (Yao et al., 2019) to assess the degree of repetition and diversity within event sequences and generated stories.

3.3 Experimental Results

Evaluation of Event Sequences As shown in Table 1, when considering all metrics, both EventAdvisor and NGEP substantially outperform the selected baselines. Performance on the referenced metrics, $ROUGE$ and $BLEU$, indicates that the events predicted by our proposed models are more similar to the human-written event sequences. We hypothesise that the superior performance of EventAdvisor over NGEP is a result of select test events not being present in G, with our event advisor being more robust to such cases.

Performance on Story Generation Table 2 measures the quality of generated stories[6] on unreferenced metrics conditioning on the leading context C and event plans E. We observe that NGEP substantially outperforms all baseline models. This indicates that our proposed graph-based inference improves story generation through planning better storylines, as our predicted events have no hallucination problems and contain event sequences that are more logically coherent. The intra-story repetitions shown in Figure 3 further demonstrate that the proposed model is more stable throughout the generation process (less fluctuations), and the predicted events display less repetition, improving the diversity of stories.

In-depth Analysis To further study how the proposed framework works during event planning, we conduct a case study as illustrated in Figure 4. Given the leading context, we can extract the con-

[5] We additionally intended to compare our model to GraphPlan (Chen et al., 2021), which also proposed the use of event graphs to improve event planning. However, we encountered difficulties in attempting to reproduce this work, e.g., the word embedding based framework only works for one-word events and there is no publicly available code.

[6] C and E are concatenated as the input of those models.

tained event ***had test***. In the event graph constructed from the training dataset, the event ***had test*** has many candidates whose conditional probabilities are calculated by the proposed NGEP. It can be observed that the event candidate ***studied*** has the highest probability. This is because, in the training dataset, more stories contain the content "people studied hard to prepare for this test". This indicates that instead of implicitly capturing the relatedness between events through neural models, NGEP allows the predicted events to have more knowledge grounding. Therefore, compared to traditional neural event planning methods, the processes behind NGEP are easier to interpret, whilst also avoiding the hallucination problem of deep learning.

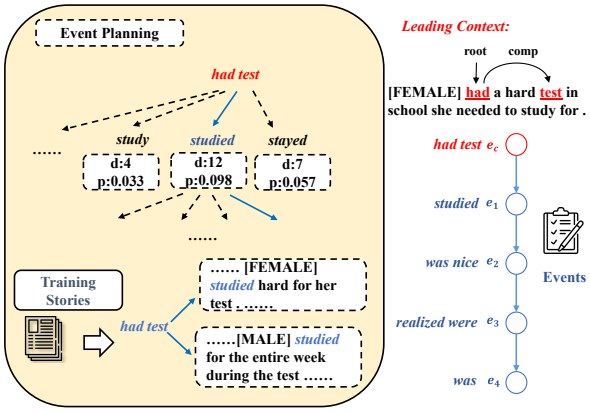

Figure 4: An example of the event planning process within our proposed NGEP. d denotes degree, and p denotes the conditional probability.

4 Conclusion

This study proposes a novel hybrid event planning approach which performs inference on event graphs with the help of a neural event advisor. A range of experiments demonstrate that the proposed model outperforms other SOTA neural event planning approaches, and substantially improves performance on the downstream task of story generation.

Acknowledgements

Chen Tang is supported by the China Scholarship Council (CSC) for his doctoral study (File No.202006120039). Tyler Loakman is supported by the Centre for Doctoral Training in Speech and Language Technologies (SLT) and their Applications funded by UK Research and Innovation [grant number EP/S023062/1]. We also gratefully acknowledge the anonymous reviewers for their insightful comments.

References

Emily Ahn, Fabrizio Morbini, and Andrew Gordon. 2016. Improving fluency in narrative text generation with grammatical transformations and probabilistic parsing. In *Proceedings of the 9th International Natural Language Generation conference*, pages 70–73, Edinburgh, UK. Association for Computational Linguistics.

Arwa I Alhussain and Aqil M Azmi. 2021. Automatic story generation: a survey of approaches. *ACM Computing Surveys (CSUR)*, 54(5):1–38.

David Bamman and Noah A. Smith. 2014. Unsupervised discovery of biographical structure from text. *Transactions of the Association for Computational Linguistics*, 2.

Jari Björne and Tapio Salakoski. 2018. Biomedical event extraction using convolutional neural networks and dependency parsing. In *Proceedings of the BioNLP 2018 workshop*, pages 98–108, Melbourne, Australia. Association for Computational Linguistics.

Hong Chen, Raphael Shu, Hiroya Takamura, and Hideki Nakayama. 2021. GraphPlan: Story generation by planning with event graph. In *Proceedings of the 14th International Conference on Natural Language Generation*, Aberdeen, Scotland, UK. Association for Computational Linguistics.

Yi Cheng, Siyao Li, Bang Liu, Ruihui Zhao, Sujian Li, Chenghua Lin, and Yefeng Zheng. 2021. Guiding the growth: Difficulty-controllable question generation through step-by-step rewriting. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5968–5978.

Marie-Catherine De Marneffe and Christopher D Manning. 2008. Stanford typed dependencies manual. Technical report, Technical report, Stanford University.

Henry Elder, Alexander O'Connor, and Jennifer Foster. 2020. How to make neural natural language generation as reliable as templates in task-oriented dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2877–2888, Online. Association for Computational Linguistics.

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, Online. Association for Computational Linguistics.

Jian Guan, Fei Huang, Zhihao Zhao, Xiaoyan Zhu, and Minlie Huang. 2020. A knowledge-enhanced pretraining model for commonsense story generation. *Transactions of the Association for Computational Linguistics*, 8:93–108.

Jian Guan, Xiaoxi Mao, Changjie Fan, Zitao Liu, Wenbiao Ding, and Minlie Huang. 2021. Long text generation by modeling sentence-level and discourse-level coherence. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6379–6393, Online. Association for Computational Linguistics.

Harsh Jhamtani and Taylor Berg-Kirkpatrick. 2020. Narrative text generation with a latent discrete plan. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3637–3650, Online. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Lara Martin, Prithviraj Ammanabrolu, Xinyu Wang, William Hancock, Shruti Singh, Brent Harrison, and Mark Riedl. 2018. Event representations for automated story generation with deep neural nets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Anna Rohrbach, Lisa Anne Hendricks, Kaylee Burns, Trevor Darrell, and Kate Saenko. 2018. Object hallucination in image captioning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Delia Rusu, James Hodson, and Anthony Kimball. 2014. Unsupervised techniques for extracting and clustering complex events in news. In *Proceedings of the Second Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, pages 26–34, Baltimore, Maryland, USA. Association for Computational Linguistics.

Chen Tang, Frank Guerin, Yucheng Li, and Chenghua Lin. 2022. Recent advances in neural text generation: A task-agnostic survey. *arXiv preprint arXiv:2203.03047*.

Lin Wang, Juntao Li, Rui Yan, and Dongyan Zhao. 2020. Plan-CVAE: A planning-based conditional variational autoencoder for story generation. In *Proceedings of the 19th Chinese National Conference on Computational Linguistics*, pages 892–902, Haikou, China. Chinese Information Processing Society of China.

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*.

A Appendix

A.1 Details of Event Schema

An event is intended to represent an important change that happens within a narrative, and so generally represents an action. The schema for an event aims to include all relevant roles to the action (e.g., verbs and object) and filter trivial details for representation. Inspired by the work of Rusu et al. (2014) and Björne and Salakoski (2018) which used dependency parsing to capture dependencies between words belonging to different clauses, we extract event mentions from sentences according to the hierarchy of typed dependencies (De Marneffe and Manning, 2008) (see details in Appendix. A.1). In this way we can obtain more informative and

unambiguous events compared to single-verb representations used in previous work (Jhamtani and Berg-Kirkpatrick, 2020; Guan et al., 2020). The schema is shown in Figure 5.

Attributes	Dependencies		Examples
Trigger	root		the predicate e.g. **drive**
Arguments	Role=modifier	prt, neg	Bill does not **drive** (neg)
	Role=agent	agent	**killed** by the crime (agent)
	Role=comp	dobj, acomp, ccomp, xcomp	**gave** me a raise (comp)

Figure 5: The schema of event shows the relations with event arguments and word dependencies. We offer some examples to indicate these dependencies, e.g., in "Bill does not drive", "not" is a negation (**neg**) of "drive", so it is an event modifier.

As shown in Figure. 5, event arguments are extracted according to selected dependencies between words. Here, we give the details of these dependencies, and Table. 3 indicates the roles of these dependencies in a sentence (for more details of dependencies see De Marneffe and Manning (2008)).

Dep.	Full Name	Example
prt	phrasal verb particle	[shut]-*prt*->[down]
neg	negation modifier	[drive]-*neg*->[not]
agent	agent	[killed]-*agent*->[police]
dobj	direct object	[gave]-*dobj*->[raise]
acomp	adjectival complement	[looks]-*acomp*->[beautiful]
ccomp	clausal complement	[says]-*comp*->[like]
xcomp	open clausal complement	[like]-*xcomp*->[swim]

Table 3: Details of dependencies in Event Schema. Examples are extracted with the format [head]-*dependency*->[tail].

The schemas of events are required to consider performance with respect to both generalisation and representation. The more dependencies included, the more potentially informative an event may become, at the cost of reduced generalisation. For instance, the *Subject* (e.g. I, you, Kent, etc.) is useful to identify the protagonist of an event, but stories usually have different characters, making it challenging to reuse events from one story in another. For example, "Kent is driving" and "He is driving" refer to the same semantic event, but if "Kent" is extracted as an event unit, it is very hard to predict the same event for another story, which means generalisation is impaired. According to a similar criterion, we select key roles as the arguments of events with the consideration of both generalisation and representation.

A.2 Details of Event Extraction

We extract events from the text of the training dataset including reference stories and leading contexts. The data structure of an event is a set including the relevant triggers and arguments in a sentence. We firstly use $spaCy$ to parse dependencies between words in a sentence, and then annotate the event trigger and arguments according to their dependencies. An event e contains attributes introduced in Figure 5, in which the event trigger is usually the predicate. Before encoders accept text as the input, the extracted events are serialised to text format to pass to the model.

Since existing story datasets do not have the reference storylines paired with reference stories, we develop an event extractor that extracts event sequences from reference stories to act as the storylines. We follow the approach of representing events as verb phrases. Verbs, as the anchor of sentences, can be seen as the *event trigger*, so our primary goal is to extract all key roles (as *event arguments*) related to the event trigger. The neighbourhood of extracted events will be considered as temporal relations.

With the temporally related events from the training stories, we construct an event graph denoted G, which is an isomorphic graph with a single event type and a single relation type. We suppose G is a data structure composed of triples in e_h, r, e_t format. The workflow of the extraction process is explained as follows:

Algorithm 2: Extract Event Sequence E

Input: A story S with m sentences
Output: Event Sequence E for S
containing m event objects

Initialise $E \leftarrow \varnothing$ and
$roles \leftarrow \{trigger, mod, agent, comp\}$
foreach s^i *in* S **do**
    Initialise $e_i \leftarrow \varnothing$
    Normalise s^i and get dependencies dep_i
    with $spaCy$
    Extract event trigger t and position p_t
    from dep_i
    $e_i.trigger \leftarrow t$
    foreach $role$ *in* $role$ **do**
        if $t \in dep_i.heads$ *and*
        $role \in dep_i.tails$ **then**
            Extract $(role, p_r)$ from dep_i
            $e_i.role \leftarrow (role, p_r)$
    $e_i.string \leftarrow r \in roles$ aligned by $p_r \uparrow$
    E append e_i