EtriCA: Event-Triggered Context-Aware Story Generation Augmented by Cross Attention

Chen Tang[1], Chenghua Lin[2*], Henglin Huang[1], Frank Guerin[1] and Zhihao Zhang[3]
[1]Department of Computer Science, The University of Surrey, UK
[2]Department of Computer Science, The University of Sheffield, UK
[3]School of Economics and Management, Beihang University, Beijing, China
`{chen.tang,hh01034,f.guerin}@surrey.ac.uk`
`c.lin@sheffield.ac.uk, zhhzhang@buaa.edu.cn`

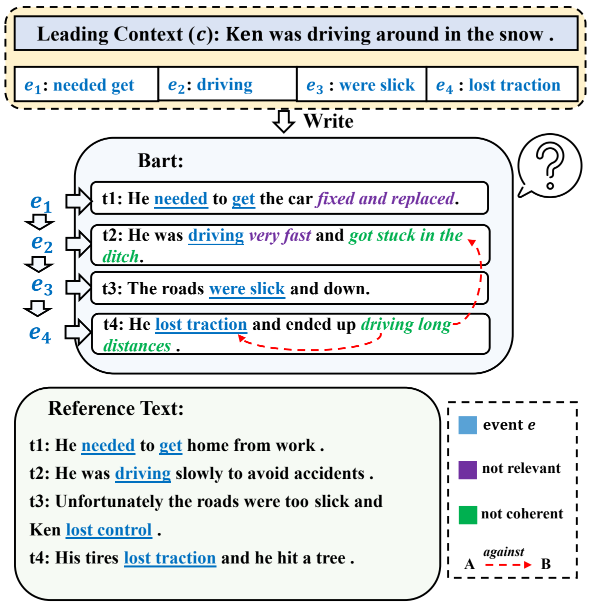
Abstract

One of the key challenges of automatic story generation is how to generate a long narrative that can maintain fluency, relevance, and coherence. Despite recent progress, current story generation systems still face the challenge of how to effectively capture contextual and event features, which has a profound impact on a model's generation performance. To address these challenges, we present EtriCA, a novel neural generation model, which improves the relevance and coherence of the generated stories through residually mapping context features to event sequences with a cross-attention mechanism. Such a feature capturing mechanism allows our model to better exploit the logical relatedness between events when generating stories. Extensive experiments based on both automatic and human evaluations show that our model significantly outperforms state-of-the-art baselines, demonstrating the effectiveness of our model in leveraging context and event features.

1 Introduction

Story Generation aims to generate fluent, relevant and coherent narratives conditioned on a given context. As the task is notoriously difficult, a common strategy is to employ storylines composed of events to support the generation process (Yao et al., 2019; Chen et al., 2021; Alhussain and Azmi, 2021; Tang et al., 2022b). This process imitates the behavior of human writers. Firstly, a story will start from a sketch of key words containing events, and then human writers will unfold the story following the track of planned event sequences.

Despite recent progress, existing approaches are still ineffective in exploiting planned events when generating stories. Usually, pre-trained generation models, e.g., BART (Goldfarb-Tarrant et al., 2020; Clark and Smith, 2021; Huang et al., 2022) are employed to generate stories after event planning. However, as shown by the conflicts in Figure 1, the separate sentences generated by BART look reasonable,



t1: He needed to get the car *fixed and replaced*.
t2: He was driving *very fast* and *got stuck in the ditch*.
t3: The roads were slick and down.
t4: He lost traction and ended up *driving long distances* .

Reference Text:
t1: He needed to get home from work .
t2: He was driving slowly to avoid accidents .
t3: Unfortunately the roads were too slick and Ken lost control .
t4: His tires lost traction and he hit a tree .

Figure 1: Conditioned on leading context and reference events (extracted from reference stories), existing generation models still suffer from problems of relevance and coherence. For instance, we fine-tune BART (Lewis et al., 2020) to generate stories. The leading context and reference text in this example are collected from ROC Stories (Mostafazadeh et al., 2016). Some conflicts among them are observed and coloured.

but there are several issues observed considering the whole story: As a commonsense story, if the car needs to be *"fixed and replaced"* then it is too broken to *"drive around"*; *"Ken"* should not drive the car *"very fast"* in the *"snow"*; If *"Ken" "got stuck in the ditch"* or *"lost traction"*, he cannot then be *"driving long distances"*. We hypothesise that these problems come from the inadequacy of capturing contextual features when keeping track of event sequences, because (i) the planned events generally lack background information, e.g., Ken (the character) and snow (the scene) and (ii) training stories may have the same events but different reference stories, which may lead to confusion during inference if not considering the story-specific scenario.

[*]Corresponding author.

Therefore, to address these challenges we propose **EtriCA** - a novel **E**vent-**Tri**ggered **C**ontext-**A**ware end-to-end framework for story generation. Given both leading context and planned events, EtriCA can more effectively capture contextual and event features from inputs than state-of-the-art (abbr. SOTA) baseline models. Traditional generation models struggle to learn contextual representations when implicitly keeping track of the state of events due to feature differences of events and contexts. As an abstract storyline, an event sequence only contains schematic information related to actions (e.g. the verb), while the context usually records story-specific details, e.g., the scene and characters in a story.

To comprehensively leverage both features, we draw inspiration from prior work dealing with information fusion (Chen et al., 2018; Xing et al., 2020; He et al., 2020; You et al., 2020; Wang et al., 2021; Tang et al., 2022a) to encode heterogeneous features with a cross attention mechanism (Gheini et al., 2021). We aim to inform our model of the context background when the neural module unfolds each event into a narrative. We propose a novel neural module that learns to implicitly map contextual features to event features through information fusion on their numeric vector spaces (we call this process contextualising events). The whole process is illustrated in Figure. 2. With the contextualised event features, an autoregressive decoder is employed to dynamically generate stories by learning to unfold the contextualised events. We also introduce an auxiliary task of Sentence Similarity Prediction (Guan et al., 2021) to enhance the coherence between event-driven sentences.

To support research on event-driven story generation, we propose a new task formulated by writing stories according to a given leading context and event sequence. We improve the event extraction framework of Chen et al. (2021) by exploiting dependency parsing to capture event related roles from sentences, instead of using heuristic rules. We also present two datasets where multi-sentence narratives from existing datasets are paired with event sequences using our automatic event extraction framework. Importantly, our task formulation can also benefit the study of controllable story generation, considering there is increasing interest in storyline-based neural generative frameworks (Xu et al., 2020; Ghazarian et al., 2021; Chen et al., 2021). According to our extensive experiments, EtriCA performs better than baseline models considering the metrics of fluency, coherence, and relevance. Our contributions[1] can be summarised as follows:

- A new task formulation for event-driven story writing, which requires the generation model to write stories according to a given leading context and event sequence.
- We annotate event sequences on two existing popular datasets for our new task, and introduce new automatic metrics based on semantic embeddings to measure the coherence and relevance of the generated stories.
- We propose a neural generation model **EtriCA**, which leverages the context and event sequence with an enhanced cross-attention based feature capturing mechanism and sentence-level representation learning.
- We conduct a range of experiments to demonstrate the advances of our proposed approach, and comprehensively analyse the underlying characteristics contributing to writing a more fluent, relevant, and coherent story.

2 Related Work

2.1 Neural Story Generation

Before the surge of deep learning techniques, story generation models only generated simple sentences and heavily relied on manual designs (McIntyre and Lapata, 2009; Woodsend and Lapata, 2010; McIntyre and Lapata, 2010; Huang and Huang, 2013; Kybartas and Bidarra, 2016). Since neural story generation came into being, end-to-end neural models, especially pre-trained models, e.g., BART (Lewis et al., 2020) and GPT-2 (Radford et al., 2019), are widely employed as the main module of story writing (Rashkin et al., 2020; Guan et al., 2020; Goldfarb-Tarrant et al., 2020; Clark and Smith, 2021). However, it is hard to guarantee logical correctness for naive Seq2Seq models when the generated text is growing longer, so recent work is exploring multi-step generations which implement neural models in traditional generative pipelines (Guan et al., 2021). For example, Yao et al. (2019); Goldfarb-Tarrant et al. (2020); Chen et al. (2021) split story generation into planning (inputs to events) and writing (events to stories), and leverage two neural generation models to learn them.

2.2 Event Planning for Story Generation

At the planning stage, prior research (Yao et al., 2019; Rashkin et al., 2020; Goldfarb-Tarrant et al., 2020;

[1]The related code is available at `https://github.com/tangg555/EtriCA-storygeneration`

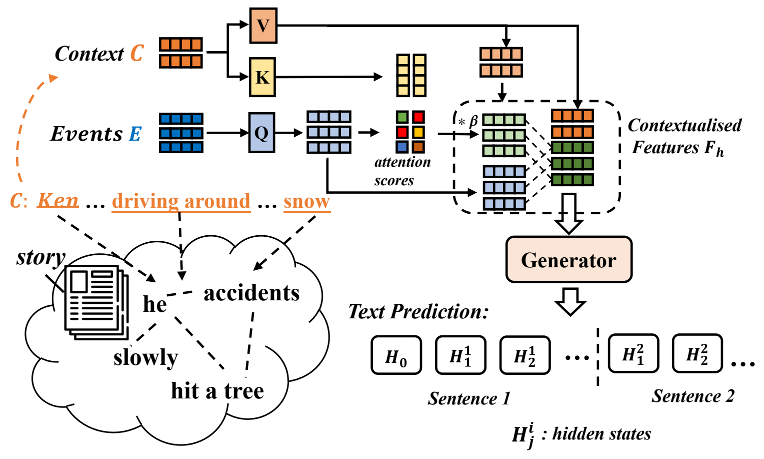
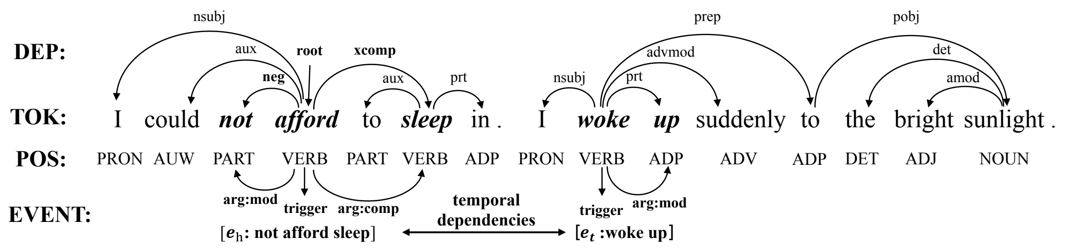


Figure 2: The overview of the event feature contextualising process. The leading context coloured in red contains some important information which affects the generation process, e.g., the weather "snows" may lead to "accident". These implicit clues help the neural generator to disambiguate the context of events. We firstly fuse both context and event features, and then feed them to the generator.



DEP:
nsubj
aux
neg
root
xcomp
aux
prt
nsubj
advmod
prt
prep
pobj
det
amod
TOK:
I could not afford to sleep in . I woke up suddenly to the bright sunlight .
POS:
PRON AUW PART VERB PART VERB ADP PRON VERB ADP ADV ADP DET ADJ NOUN
EVENT:
arg:mod
trigger
arg:comp
temporal dependencies
trigger
arg:mod
[e_h: not afford sleep]
[e_t :woke up]

Figure 3: An example to illustrate the process of event extraction. *TOK* is the basic unit of a sentence. *POS* is the part of speech, and *DEP* stands for dependencies between tokens. Through parsing dependencies, the event trigger (also recognised as the root of sentence) filters all significant roles to represent a complete action. Meanwhile, extracted neighbour events are considered to have temporal relations.

Jhamtani and Berg-Kirkpatrick, 2020; Ghazarian et al., 2021) mostly focused on extracting event sequences from the reference text as the ground truths of plot planning, and then leveraged neural models (Radford et al., 2019; Lewis et al., 2020) to predict events with given leading context or titles. Events have a lot of representation formats, e.g., verbs, tuples, key words, etc. Among them a straight forward approach is extracting verbs as events (Jhamtani and Berg-Kirkpatrick, 2020; Guan et al., 2020; Kong et al., 2021), which is also the method we followed. However, verbs alone are not good enough to keep information integrity. For instance, semantic roles like negation (not) are significant for correct understanding. Peng and Roth (2016) and Chen et al. (2021) use some heuristic rules to include these semantic roles, but those heuristic rules are not complete to include all the key roles. Therefore, inspired by related works (Rusu et al., 2014; Björne and Salakoski, 2018; Huang et al., 2018) in open-domain event extraction, we propose an event extraction workflow based dependency parsing to capture essential components for verb phrases in sentences as events.

3 Methodology

3.1 Task Formulation

Under the umbrella of controllable story generation we define the following task: write a story that leverages both the given leading context and a given planned event sequence. Our primary goal is to investigate how to consider the context while keeping track of the given event sequence with neural generation models, so we expand the original context-aware story generation settings of Guan et al. (2021) by adding an event sequence, for each leading context, as the storyline to follow.

Input: The input for each sample includes a leading context $C = \{c_1, c_2, ..., c_n\}$ which acts as the first sentence of a story, and an event sequence $E=\{e_1,e_2,...,e_m\}$ as a storyline to build up a sketch for a story. c_i means the i-th token of the leading context, and e_i means the i-th event representing the i-th sentence in a story.

Output: The output is a multi-sentence story $S = \{s_1^1, s_2^1, ..., s_1^2 ... , s_n^m\}$. , s_j^i denotes the j-th token of i-th sentence in a story.

3.2 Event Sequence Preparation

Following the prior work of event extraction (Chen et al., 2021) (discussed in Sec. 2.2), we present an automatic framework which includes all verb related roles by analysing dependencies between words[2]. The event representation is not the focus of this study however. Figure 3 shows an example of the event extraction process. The details of the event schema are appended in Appendix. A.2.

3.3 Neural Framework

At the writing stage, conditioned on a leading context C and planned events E (see Sec. 3.1), the neural model generates a multi-sentence story S. Figure 4 shows the overview of the whole process.

Planned Events Representation The language models of conventional generation frameworks either based on transformers (Vaswani et al., 2017) or RNNs (Ghosh et al., 2017), are commonly designed to encode natural language input text, so the extracted events need to be firstly serialised to plain text. We separate the string format of events introduced in Sec. 3.2 with special tokens, e.g., "<e_s> needed get <e_sep> ... <e_e>", where "<e_s>","<e_sep>", "<e_e>" represent the start, separation, and end of planning, respectively.

Contextualised Features Representation At the encoding stage, neural models have the inputs of C and E which have feature differences as mentioned above. Conventional end-to-end models usually concatenate embeddings of different inputs, since neural encoders capture their features in numeric vector space (e.g. through self-attention). However, when the event sequence grows longer, the growing concatenated embeddings may decrease the influence of C (we discuss this in Appendix A.4). Instead, we firstly leverage two separate BART (Lewis et al., 2020) encoders to incorporate features, and then fuse features with multi-head attentions calculated as below:

$$F_c = \text{Encoder}_c(C); F_e = \text{Encoder}_e(E) \tag{1}$$

$$Q_i = W_i^Q F_e, K_i = W_i^K F_c, V_i = W_i^V F_c, \tag{2}$$

$$A_i = \text{softmax}(\frac{Q_i K_i^{\mathrm{T}}}{\sqrt{d_k}}) V_i \tag{3}$$

$$F_{ca} = \text{Concat}(A_1, ..., A_m) W^M \tag{4}$$

where $Encoder_c$ and $Encoder_e$ inherit pre-trained parameters from BART but do not share trainable parameters when fine-tuning. F_c and F_e stand for the features captured from C and E, respectively. i denotes the i-th head of the attention scores which have m heads in total. W_i^Q, W_i^K, W_i^V, and W^M are trainable parameters. The i-th head attention A_i is the attention-based weight sum of the feature matrix. Finally, the obtained F_{ca} represents the attentions of ongoing events under the consideration of context.

In order to contextualise the input event features, we incrementally add the F_{ca} to the original event features F_e so that neural models are forced to learn the context gap between event sequences and stories, i.e.,

$$F_{he} = F_e + \beta \odot F_{ca} \tag{5}$$

$$F_h = \text{Concat}(F_c, F_{he}) \tag{6}$$

where β denotes the scale factor of F_{ca}. $\beta \odot F_{ca}$ is the representation of the context gap trained via residual mapping. F_h concatenates both leading context features F_c and contextualised event features F_{he}. These are fed into a neural decoder to predict tokens and sentence representations.

Decoding and Sentence-level Fitting As in other conventional generation systems, we employ an auto-regressive decoder to generate story tokens y_t as equations below.

$$H_t = \text{Decoder}(y_{<t}, F_h) \tag{7}$$

$$P(y_t | y_{<t}, X) = \text{softmax}(H_t W) \tag{8}$$

$$y_t \overset{sampling}{\longleftarrow} P(y_t | y_{<t}, F_h) \tag{9}$$

where t denotes a time step. X denotes the input to the neural model. H_t is the t-th hidden state of the decoder module. H_t is computed from the information of both context and events contained in F_h and the prior predicted story $y_{<t}$. W is a trainable parameter, and $P(y_t | y_{<t}, F_h)$ is the probability distribution of the vocabulary including special tokens. Through a sampling strategy, e.g., *argmax*, we collect the predicted token y_t.

[2]We use *spaCy* (`https://spacy.io/`) to split sentences and parse dependencies between words in each sentence.

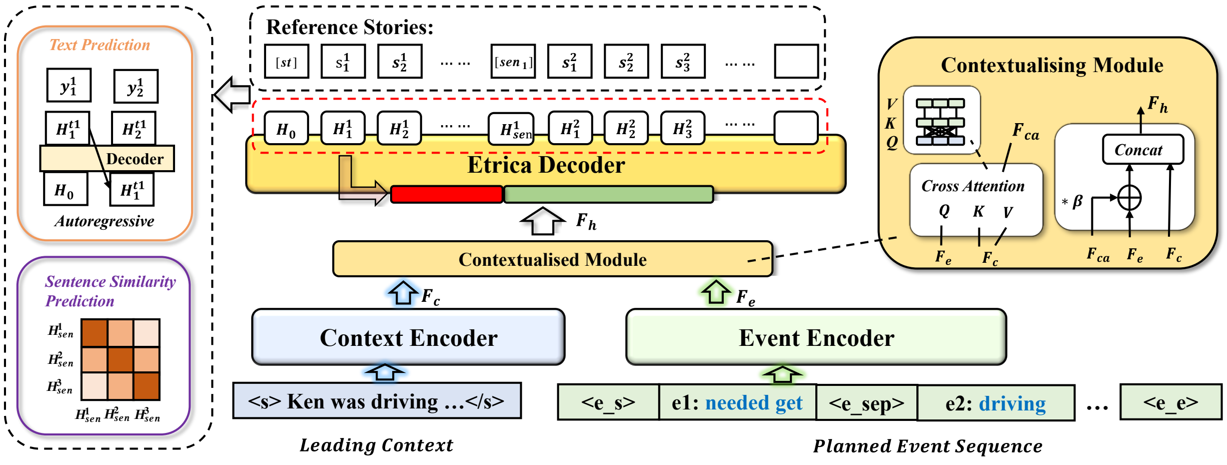


Figure 4: The overview of EtriCA architecture. The technique details are explained in Sec 3.3. When training, in addition to predicting text tokens $\{y_1^1, ..., y_j^i\}$ one by one, we train the decoder to learn sentence-level representations by the similarity prediction auxiliary task shown in the dotted box. Through representation learning, neural models learn how to generate the reference-like stories with given leading context and planned event sequence.

In addition to the token-level representation, we introduce an auxiliary task of Sentence Similarity Prediction (Guan et al., 2021) to learn sentence-level representations and training methods. Due to the page limit, the details are moved to Appendix A.4.

Training and Inference As shown in Figure 4, the neural model is trained to fit both token and sentence level references as follows:

$$\mathcal{L}_{lm} = -\frac{1}{N}\sum_{t=1}^{N} logP(y_t|y_{<t}, X) \tag{10}$$

$$\mathcal{L}_{sent} = \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m}(max|sim_{ij}^s - sim_{ij}^y|, \Delta) \tag{11}$$

$$\mathcal{L}_{overall} = \mathcal{L}_{lm} + \lambda\mathcal{L}_{sent} \tag{12}$$

where $\mathcal{L}_{lm}$ is the cross-entropy loss of $P(y_t|y_{<t}, F_h)$. $\mathcal{L}_{sent}$ is the loss of predicted sentence similarities. sim_{ij}^s and sim_{ij}^y denote the sentence similarities between the i-th and j-th sentences in a reference story and a generated stories, respectively. λ is an adjustable scale factor, and $\mathcal{L}_{overall}$ is the overall loss. By minimising $\mathcal{L}_{overall}$, the neural model learns to predict a human-like story. The Sentence Similarity Prediction only works on training, and when doing inference the neural model finally outputs stories without those special tokens.

4 Experiment

4.1 Datasets

In this study, we annotate two popular datasets, ROCStories (ROC) (Mostafazadeh et al., 2016) and Writing Prompts (WP) (Fan et al., 2018) with extra event sequences as our benchmarks. We follow the settings of prior work (Xu et al., 2020; Guan et al., 2021) to preprocess these data. The stories in both datasets are split into sentences by NLTK (Bird and Loper, 2004). The data of ROC are delexicalised by masking all the names with tokens of *[MALE]*, *[FEMALE]*, and *[NEUTRAL]*. The data of WP are recollected from the original development and test set, and we retain the first eleven sentences in each story[3]. For both datasets, the first sentence in a story is extracted to be the leading context C as the input, and the rest is used as the reference story S. Finally, we obtain a long story dataset, WP (10 sentences), and a short story dataset, ROC (4 sentences) for the following experiments. The event sequence E is extracted from the reference story S as the planned plot to guide story generation. Statistically, the ROC has stories as the Train/Dev/Test set of 88344/4908/4909 stories, respectively, and the split of WP is 26758/2000/2000.

4.2 Baselines

We compare EtriCA with following SOTA generation models: (1) **P&W** (Plan and Write) (Yao et al., 2019): The main architecture is based on a BiLSTM with an attention mechanism (Garg et al., 2019). To make the comparison more fair, we enhance the original code by replacing original static word embeddings with the dynamic embeddings of the pre-trained BART; (2) **GPT-2** (Radford et al., 2019): A popular auto-regressive generative model which has been

[3]The original WP dataset is too large, and the topics are unconstrained.

Models	ROC Stories						Writing Prompts					
	PPL↓	R-1↑	R-2↑	R-L↑	B-1↑	B-2↑	PPL↓	R-1↑	R-2↑	R-L↑	B-1↑	B-2↑
P&W$_{l+e}$	6.22	22.82	2.65	15.90	0.297	0.150	16.47	23.49	1.74	12.17	0.259	0.086
GPT-2$_{l+e}$	10.02	29.85	6.45	20.58	0.347	0.201	49.48	18.59	2.18	10.38	0.130	0.051
BART$_{l+e}$	3.39	48.74	21.95	40.69	0.505	0.351	10.84	37.19	8.14	22.73	0.351	0.174
HINT$_{l+e}$	3.97	46.71	20.81	37.21	0.488	0.337	14.45	38.86	8.98	23.06	0.373	0.190
EtriCA (ours)	**2.88**	**49.29**	**22.59**	**41.43**	0.506	0.354	**8.11**	**39.90**	**9.65**	**25.21**	**0.387**	**0.202**
- w/o sen	3.33	49.18	22.39	41.09	**0.512**	**0.359**	9.88	39.88	9.37	24.86	0.385	0.199
- w/o cm	2.97	48.53	21.55	40.34	0.499	0.345	9.15	36.08	7.55	21.01	0.356	0.175
- w/o leading	3.24	42.55	17.21	35.90	0.450	0.287	9.37	35.46	7.22	20.69	0.357	0.172
- w/o events	4.50	24.51	2.70	16.86	0.311	0.156	12.77	23.77	1.89	12.26	0.263	0.089

Table 1: Automatic evaluation of referenced metrics on ROC and WP datasets. The best performance in each line is highlighted in **bold**. ↑/↓ means the higher/lower the better, respectively. $_{l+e}$ means the input of the model concatenates the leading context and event sequence. **w/o sen**, **w/o cm**, **w/o leading**, and **w/o events** means ablating the auxiliary task of sentence similarity prediction, respectively, the contextualising module, the leading features, and the event features.

widely used in prior works (Rashkin et al., 2020; Guan et al., 2020; Clark and Smith, 2021); (3) **BART**: This is a composed model constructed with a BERT-like (Devlin et al., 2019) encoder and a GPT-like decoder, and shown advances in prior NLG works (Goldfarb-Tarrant et al., 2020; Clark and Smith, 2021). (4) **HINT** (Guan et al., 2021): It is currently the SOTA framework on context-aware story generation, which enhanced the coherence and relevance through training with two training objectives.

4.3 Implementation Details

The main contribution of our generation model is the contextualising module, which can adapt to other encoder-decoder frameworks. Therefore, we employ the encoders and decoders from the BART framework (Lewis et al., 2020), which has shown strong performance in prior studies (Goldfarb-Tarrant et al., 2020; Guan et al., 2021), to build our neural generation model. We fine-tune our generation model based on a publicly available BART checkpoint[4] and fix the random seed to 42. All of our code is implemented in PyTorch, and trained with the PyTorch Lightning framework. More details of the hyper-parameters of the model, training and inference are described in Appendix A.1.

4.4 Automatic Evaluation

4.4.1 Evaluation Metrics

Perplexity (PPL) measures the uncertainty of generated tokens predicted by neural models. **ROUGE-n (R-n)** (Lin, 2004) is a set of reference metrics measuring the coverage rate between generated stories and the referenced stories where n denotes n-grams. **BLEU-n (B-n)** (Papineni et al., 2002) is also a set of reference metrics to compute n-gram overlaps between the generated stories and the references. **Lexical Repetition-n (LR-n)** is an unreferenced metric to compute the percentage of generated stories which have a 4-gram repeated at least n times (Shao et al., 2019). **Distinction-n (D-n)** is an unreferenced metric qualifying the distinction of stories by measuring the ratio of distinct n-grams to all those generated n-grams (Li et al., 2016). **Intra-story Repetition** (Yao et al., 2019) measures the repetition of each sentence in a story by the overlaps of trigrams. **Intra-story Coherence and Relevance** (Xu et al., 2018), originally used in dialogue evaluation, is based on cosine similarity between semantic embeddings[5] to calculate sentence-level coherence and relevance. This approach is used to measure the relatedness between neighbouring generated sentences as the intra-story coherence, and the relatedness between the leading context and story sentence as the intra-story relevance[6]. **Intra-story Aggregate Metrics** i.e. repetition, coherence, and relevance, are obtained by the mean of sentence-level metrics.

4.4.2 Evaluation Results

Reference metrics. Table 1 shows the automatic evaluation results on both the short story dataset ROC and the long story dataset WP. It can be observed that EtriCA outperforms all baselines on all metrics for both datasets. Compared to the strongest baseline BART and HINT, our model reduces perplexity by 15% on ROC and 25% on WP, respectively. For the

[4]The checkpoint of bart-base loaded from https://huggingface.co/facebook/bart-base

[5]Glove Vectors are used here. https://nlp.stanford.edu/projects/glove/

[6]This work is originally an unsurpervised metric developed for conversations. We use the same methods for evaluating our generated stories by implementing it on the sentences in a story. The code we use is from the repository https://github.com/tonywenuon/dialog-coherence-metric.

Models	ROC Stories		Writing Prompts	
	LR-2↓	D-4↑	LR-2↓	D-4↑
P&W$_{l+e}$	0.297	0.773	0.443	0.834
GPT-2$_{l+e}$	0.528	0.675	0.760	0.684
BART$_{l+e}$	0.245	**0.804**	0.378	0.894
HINT$_{l+e}$	0.264	0.734	**0.338**	0.855
EtriCA	**0.244**	0.799	0.359	0.889
- w/o sen	0.286	0.794.	0.343	**0.900**
- w/o cm	0.245	0.800	0.514	0.827
- w/o leading	0.260	0.795	0.517	0.892
- w/o events	0.245	0.792	0.412	0.850
Golden	0.048	0.906	0.286	0.950

Table 2: Automatic evaluation of unreferenced metrics on the ROC and WP datasets for the generation models writing stories conditioned on both leading text and the reference event sequence. **Golden** in the table denotes the reference stories.

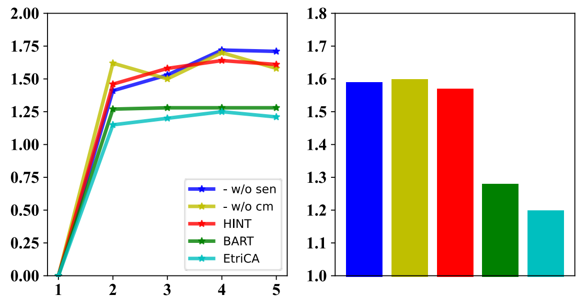


Figure 5: The results of intra-story repetitions and aggregate scores on stories of the ROC dataset. The curve graphs illustrate the intra-story repetition for each sentence (leading context as the first sentence) in a story. The histograms depict the aggregate scores of intra-story repetitions over the story sentences.

BLEU and ROUGE metrics, EtriCA also outperforms other baselines, which demonstrates that EtriCA generates stories more closely resembling the human written reference stories.

In addition, with the ablation study, it can be observed that both the context and event features play an important role in improving the generation process. Considering the performance of *- w/o leading* and *- w/o events*, they indicate that the features contained in the two kinds of inputs are complementary to each other, and both of them are essential for good story writing. Therefore, the task of how to effectively incorporate both features is important for enhancing the ability of writing a high-quality story. When EtriCA does not implement our contextualising module (abbr. cm), all the metrics substantially drop, and some metrics even become lower than those of BART$_{l+e}$ and HINT$_{l+e}$. This observation suggests that our contextualising module can more effectively fuse the heterogeneous features, and generate a richer semantic representation for the following story writing. Similarly, the sentence-level representations also improve most of the metrics, although not as much as the contextualising module. We hypothesise it is because the contextualised module has significantly reduced the gap between event sequences and the stories (each event is paired with each sentence), making the improvement by sentence-level representations less prominent. Our hypothesis is also confirmed in the following experiments.

Unreferenced Metrics. In another set of experiments, we examine repetition and diversity of the generated stories, where the results are reported in Table 2. It can be observed that EtriCA gives strong performance for both lexical repetition (LR-2) and diversity (D-4), either achieves the best performance or is on a par with the best performing baseline. To further investigate how our model performs on writing along with the planned events, we follow Yao et al. (2019) to observe the intra-story repetitions for each generated sentence, as shown in Figure 5. The results show that EtriCA consistently outperforms the baselines for both sentence-level and story-level (i.e., aggregated) repetitious scores, indicating that EtriCA performs better on event-triggered story writing.

In-depth analysis Furthermore, we present more experimental results[7] to analyse the intra-story coherence and relevance of our models. We select the two strongest baselines according to previous experimental results to compare. As shown in Table 3, our approach consistently outperforms the baselines for both the intra-story coherence and relevance. This indicates our contextualising module improves the feature capturing for both the context and event features to enhance the logical relatedness between story sentences, and between the story and the input.

Additionally, the ablation results, in which EtriCA and "- w/o sen" have very close performance, also confirm the aforementioned hypothesis that the feature capturing mechanism of the contextualising

[7]We tried to conduct intra-story analysis (repetition, coherence, relevance) on the WP dataset. However, even in the referenced stories, there are a great number of sentences that are too short and conversational, e.g. "well ?" or "you said that ?", to be meaningfully analysed. Therefore, we only conduct intra-story experiments on the ROC dataset.

Models	Coherence			Relevance		
	wiki.	twit.	comm.	wiki.	twit.	comm.
BART$_{l+e}$	0.4658	0.6293	0.5865	0.5316	0.6710	0.6439
HINT$_{l+e}$	0.4627	0.6276	0.5818	0.5323	0.6718	0.6427
EtriCA	0.4667	0.6306	**0.5876**	0.5332	0.6722	0.6445
- w/o sen	**0.4680**	**0.6322**	0.5864	**0.5356**	**0.6745**	**0.6457**
- w/o cm	0.4602	0.6232	0.5775	0.5281	0.6676	0.6381
Golden	0.6631	0.7996	0.8298	0.6610	0.7997	0.8265

Table 3: The results of aggregate scores of intra-story coherence and relevance for the ROC dataset, which are calculated based on semantic embeddings. **wiki.**, **twit.**, **comm.** denotes the glove embeddings of "Wikipedia 2014 + Gigaword 5 (6B tokens)", "Twitter (2B tweets, 27B tokens)", and "Common Crawl (42B tokens)", respectively.

module partly replaces the functions of sentence-level representations.

Figure 6 further shows that our model significantly outperforms baselines for coherence and relevance, which indicates its advance in feature capturing and generating more event and context related stories.

Leading Context: [MALE] had lost his dog over a month ago . **Event Sequence:** missed dog → notices something → sees dog → turns out be
BART$_{l+e}$**:** He **missed** his **dog** for a whole month. One day he **notices something** moving and is startled. He **sees** the **dog** on the floor. It **turns out** to **be** a squirrel.
HINT$_{l+e}$**:** One day [MALE] **missed** his **dog**. He **notices something** about her name on the dog's tag. [MALE] **sees** the **dog** in the tags. It **turns out** it could **be** a dog from the police department.
EtriCA: He **missed** his **dog** badly. He **notices something** strange on the curb. He **sees** the **dog** outside. It **turns out** to **be** a stray dog.
- w/o sen: He had **missed** his **dog** so much that he had to search for him. As he was searching, he **notices something** about a dog. He **sees** the **dog** with a bag. It **turns out** to **be** a stray, a wad of dog spray.
- w/o cm: [MALE] **missed** his **dog** this summer. He **notices something** on his neighbor's wall about a house. [MALE] notices the dog was very sad. it **turns out** that there must **be** a really sad day next time.

Table 4: A case study for an example in **ROC Stories**. *[MALE]*, *[FEMALE]*, and *[NEUTRAL]* are the specital tokens to replace names in stories. The highlighted bold words denote the events corresponding to the given event sequence.

4.5 Human Evaluation

We conducted human evaluation based on pair-wise comparisons with two competitive baselines and the ablated model without our proposed contextualising module. We randomly sample 150 stories from the test datasets of ROC Stories[8]. There are 3 evaluators invited to choose which generated story is better (Win/Lose/Tie) on three aspects: (i) **Fluency** considers each sentence in isolation and measures the quality from a linguistic perspective, e.g., the grammatical correctness or the correct representation of semantic meaning; (ii) **Coherence** measures the logical relatedness between story sentences; (iii) **Relevance** measures the context relevance between stories and the leading contexts. When summarising the human annotation result, the final results are counted by majority voting.

Choices(%)	**Etri.** vs. ***w/o cm***		
	Etri.	***w/o cm***	**Kappa**
Fluency	**36.1****	18.0	55.3
Coherence	**40.2****	22.7	48.9
Relevance	**23.2**	21.7	48.5
Choices(%)	**Etri.** vs. **BART**$_{l+e}$		
	Etri.	**BART**$_{l+e}$	**Kappa**
Fluency	**33.6****	16.4	56.2
Coherence	**32.8***	19.1	48.1
Relevance	**16.8**	9.8	45.1
Choices(%)	**Etri.** vs. **HINT**$_{l+e}$		
	Etri.	**HINT**$_{l+e}$	**Kappa**
Fluency	**32.3****	17.3	55.4
Coherence	**35.3***	21.8	58.3
Relevance	**14.9**	8.5	50.9

Table 5: Human evaluation results on the ROC dataset. The scores stand for the percentage of model chosen in pair comparisons (win another model). Kappa means the Fleiss' Kappa (Fleiss, 1971) coefficient that is used to measure inter-annotator agreement. All of our results have reached moderate agreement. $*$ refers to significance at p<0.05, whilst $**$ refers to significance at p<0.01, on a sign test.

As shown in Table 5, EtriCA outperforms SOTA baselines in terms of fluency, coherence, and relevance. All generation models have relatively little conflict with the given input, so they all have good performance on relevance, causing less differences in relevance. On the other hand, the advances on fluency and coherence are very significant, indicating the advantage of our contextualising module in capturing high-level features from the context and event sequences.

4.6 Case Study

As is shown in Table 4, examples indicate that compared to baseline models, EtriCA generates a

[8]WP is not used to conduct experiments since long stories are difficult to get acceptable annotation agreement.

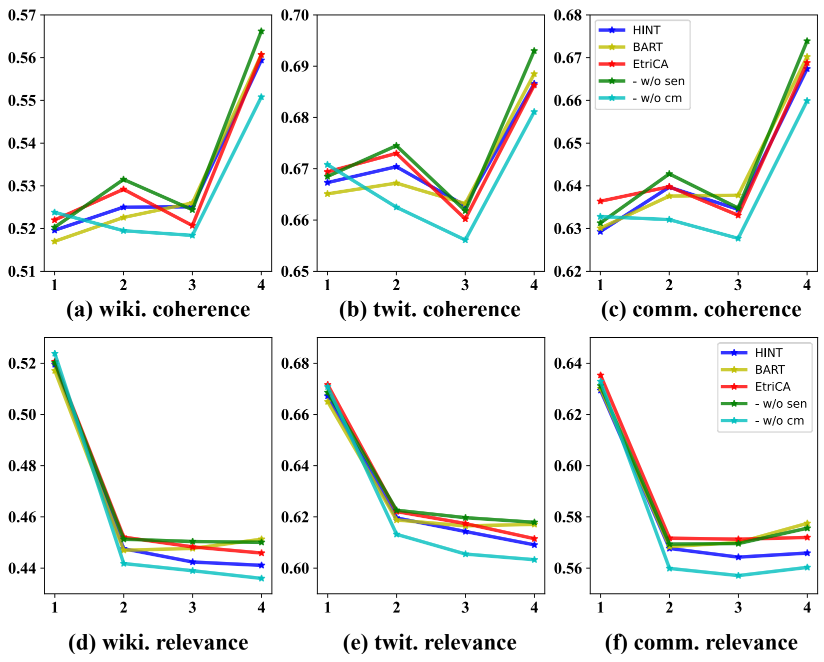


Figure 6: The results of intra-story coherence and relevance on the ROC dataset.

more logically related story with given inputs, and contains fewer confusing expressions inside the story. (More examples in Appendix. A.6)

5 Conclusion

We present a controllable story generation task conditioned with leading contexts and event sequences. We also provide two new datasets with extra annotated events, and introduce a new set of automatic metrics to measure coherence and fluency. We propose EtriCA, a novel generation model, which better exploits context and event features with a cross attention based contextualising network. Through extensive experiments and analyses, we demonstrate EtriCA can generate stories with better fluency, coherence, and relevance compared to competitive baselines.

Acknowledgements

Chen Tang is supported by the China Scholarship Council (CSC) for his doctoral study (File No.202006120039). We also gratefully acknowledge the anonymous reviewers for their insightful comments.

Limitations

According to our methodology and some observations from experiments, we conclude the limitations around our study as follows:

- **Robustness:** The quality of given events may affect the robustness of generation models. It is very hard to define whether a plot (an event sequence) is "interesting" or "bad". When provided with unusual or strange events for a story, the generation model struggles to follow the events cohesively.
- **Sequence Length:** Our neural models cannot write stories that are too long in length. Because of the features of neural encoders and decoders, the inputs and outputs generally have a certain length limitation, e.g. 1024 tokens.
- **Generalisation:** The training datasets of stories limit the topics of open domain story writing. Although large-scale pre-training has been widely adopted, our neural models still struggle to tackle some topics, writing styles, etc. For instance, the performance of story generation on WP is relatively worse than on ROC, because WP contains spoken language and irregular expressions.
- **Experiments:** For the limitation of resources, we did not conduct some further experiments in this study. For instance, we did not further study the performance of difference event representations because it is not the focus of this study.

References

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

Steven Bird and Edward Loper. 2004. NLTK: The natural language toolkit. In *Proceedings of the ACL Interactive*

Poster and Demonstration Sessions, pages 214–217, Barcelona, Spain.

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.

Di Chen, Jiachen Du, Lidong Bing, and Ruifeng Xu. 2018. Hybrid neural attention for agreement/disagreement inference in online debates. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium.

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.

Elizabeth Clark and Noah A. Smith. 2021. Choose your own adventure: Paired suggestions in collaborative writing for evaluating story generation models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3566–3575, Online.

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

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, Minneapolis, Minnesota.

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.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Sarthak Garg, Stephan Peitz, Udhyakumar Nallasamy, and Matthias Paulik. 2019. Jointly learning to align and translate with transformer models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4453–4462, Hong Kong, China.

Sarik Ghazarian, Zixi Liu, Akash S M, Ralph Weischedel, Aram Galstyan, and Nanyun Peng. 2021. Plot-guided adversarial example construction for evaluating open-domain story generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Online.

Mozhdeh Gheini, Xiang Ren, and Jonathan May. 2021. Cross-attention is all you need: Adapting pretrained transformers for machine translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 1754–1765. Association for Computational Linguistics.

Sayan Ghosh, Mathieu Chollet, Eugene Laksana, Louis-Philippe Morency, and Stefan Scherer. 2017. Affect-LM: A neural language model for customizable affective text generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada.

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)*, Online.

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.

Xuanli He, Quan Hung Tran, Gholamreza Haffari, Walter Chang, Zhe Lin, Trung Bui, Franck Dernoncourt, and Nhan Dam. 2020. Scene graph modification based on natural language commands. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, Online.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Henglin Huang, Chen Tang, Tyler Loakman, Frank Guerin, and Chenghua Lin. 2022. Improving chinese story generation via awareness of syntactic dependencies and semantics. In *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 (AACL-IJCNLP)*, Online.

Lifu Huang and Lian'en Huang. 2013. Optimized event storyline generation based on mixture-event-aspect model. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 726–735, Seattle, Washington, USA.

Lifu Huang, Heng Ji, Kyunghyun Cho, Ido Dagan, Sebastian Riedel, and Clare Voss. 2018. Zero-shot transfer learning for event extraction. In *Proceedings* *of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2160–2170, Melbourne, Australia.

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.

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

Xiangzhe Kong, Jialiang Huang, Ziquan Tung, Jian Guan, and Minlie Huang. 2021. Stylized story generation with style-guided planning. *arXiv preprint arXiv:2105.08625*.

Ben Kybartas and Rafael Bidarra. 2016. A survey on story generation techniques for authoring computational narratives. *IEEE Transactions on Computational Intelligence and AI in Games*, 9(3):239–253.

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*, Online.

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.

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

Neil McIntyre and Mirella Lapata. 2009. Learning to tell tales: A data-driven approach to story generation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 217–225, Suntec, Singapore.

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, Uppsala, Sweden.

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.

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.

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.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 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)*, Online.

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.

Zhihong Shao, Minlie Huang, Jiangtao Wen, Wenfei Xu, and Xiaoyan Zhu. 2019. Long and diverse text generation with planning-based hierarchical variational model. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3257–3268, Hong Kong, China.

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

Chen Tang, Zhihao Zhang, Tyler Loakman, Chenghua Lin, and Frank Guerin. 2022b. NGEP: A graph-based event planning framework for story generation. In *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 (AACL-IJCNLP)*, Online.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Dingmin Wang, Chenghua Lin, Qi Liu, and Kam-Fai Wong. 2021. Fast and scalable dialogue state tracking with explicit modular decomposition. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 289–295, Online.

Kristian Woodsend and Mirella Lapata. 2010. Automatic generation of story highlights. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 565–574, Uppsala, Sweden.

Xinyu Xing, Xiaosheng Fan, and Xiaojun Wan. 2020. Automatic generation of citation texts in scholarly papers: A pilot study. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6181–6190, Online.

Peng Xu, Mostofa Patwary, Mohammad Shoeybi, Raul Puri, Pascale Fung, Anima Anandkumar, and Bryan Catanzaro. 2020. MEGATRON-CNTRL: Controllable story generation with external knowledge using large-scale language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2831–2845, Online.

Xinnuo Xu, Ondřej Dušek, Ioannis Konstas, and Verena Rieser. 2018. Better conversations by modeling, filtering, and optimizing for coherence and diversity. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3981–3991, Brussels, Belgium.

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*.

Weiqiu You, Simeng Sun, and Mohit Iyyer. 2020. Hard-coded Gaussian attention for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7689–7700, Online.

A Appendix

A.1 Implementation Details

Hyparameters of model To leverage pre-trained parameters, most of the hyparameters for the encoder, decoder, and embedding layers are restored from the public checkpoints[9] on Huggingface. In addition, we set other parameters as follows: The residual scale factor β in Equation. 5 is set to 0.1. The margin Δ in Equation. 11 is set to 0.1. The scale factor λ in Equation. 12 is set to 0.1. There are also some parameters are learnable via training on datasets. In our framework, the two encoders and the decoder all have 6 hidden layers implementing 12-head attention mechanism. Both encoders and decoders share the embeddings layer containing vocabulary up to 50,625 tokens with Byte-Pair Encoding (Radford et al., 2019), and additional special tokens mentioned in Sec. 3.

Training Settings Our experiments are carried out on multiple GPUs (e.g., RTX A4000) on a cloud platform, so for the convenience of reproduction, we fix the random seed to 42. When training neural models, we implement PyTorch Lightning[10] framework to set up training processes. The training parameters setup are listed as follows: The *batch size* is set to 64; The *learning rate* is 8e-5; *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 will last for 5 *epochs*, but the result only considers the checkpoint with the best performance on the metric of *loss* (the lowest). It is worth mentioning that EtriCA needs two separate encoders to encode context (natural language) and events (concatenated serialised events) separately, but the encoder of public BART checkpoint is only pre-trained on natural language text. Therefore, to make the event encoder learn event features better, we firstly train a BART model on stories given both the context and planned events, and then restore the pre-trained encoder paratermeters to the event encoder of EtriCA.

Inference Settings When evaluating and testing, we adopt the nucleus sampling (Holtzman et al., 2019) strategy to generate texts. We also change the *batch size* to 15 when doing the inference, because nucleus sampling requires large amount of memory,

A.2 Details of Event Schema

An event is supposed to represent an important change that happens, so generally the representation of an action. The schema for an event aims to include all relevant roles to the action and filter trivial details for representation. Inspired by the work of Rusu et al. (2014); Björne and Salakoski (2018) which used dependency parser 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.2). In this way we can obtain more informative and unambiguous events compared to single-verb events used in previous work (Jhamtani and Berg-Kirkpatrick, 2020; Guan et al., 2020; Kong et al., 2021). The schema is shown in Figure 7.

As shown in Figure. 7, event arguments are extracted according to selected dependencies between words. Here, we give the details of these dependencies, and Table. 6 indicate the roles of

[9]BART from the bart-base `https://huggingface.co/facebook/bart-base`, and GPT-2 from gpt2-base `https://huggingface.co/gpt2`

[10]It offers a lot of api interfaces which simplify engineering. `https://www.pytorchlightning.ai/`

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

Figure 7: 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**) to "drive", so it is an event modifier.

these dependencies in a sentence. (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 6: Details of dependencies in Event Schema. Examples are extracted with the format of [head]-*dependency*->[tail].

The schemas of events are required to consider performance with respect to both generalisation and representation. More dependencies included can make an event more informative but may cut down its generalisation ability. For instance, *Subject* (e.g. I, you, Kent, ...) is useful to state the character of an event, but stories usually have different characters that may make it difficult for extracted events from one story to be reused in another story. E.g., "Kent is driving" and "He is driving" are the same meaning but if the subject "Kent" is extracted as an event role, it is very hard to predict same event for another story, which means the generalisation is damaged. According to similar criterion we select key roles as the arguments to events with the consideration of both generalisation and representation.

A.3 Details of Event Extraction

We extract events from the text of training dataset including reference stories and leading contexts. The data structure of an event is a set including relevant trigger and arguments in a sentence. We firstly use *spaCy*[11] to parse dependencies between words in a sentence, and then annotate event trigger and arguments according to the dependencies. An event e contains attributes introduced in Figure 7, in which the event trigger as the root is generally the predicate. Before encoders accept text as the input, extracted events will be serialised to text format to feed the model.

Since existing story datasets do not have the reference storylines paired with reference stories, we develop an event extractor to extract event sequences from reference stories as the storylines. We follow the route to represent 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 the key roles (as *event arguments*) related to the event trigger. The neighbourhood of extracted events will be considered as temporal relations.

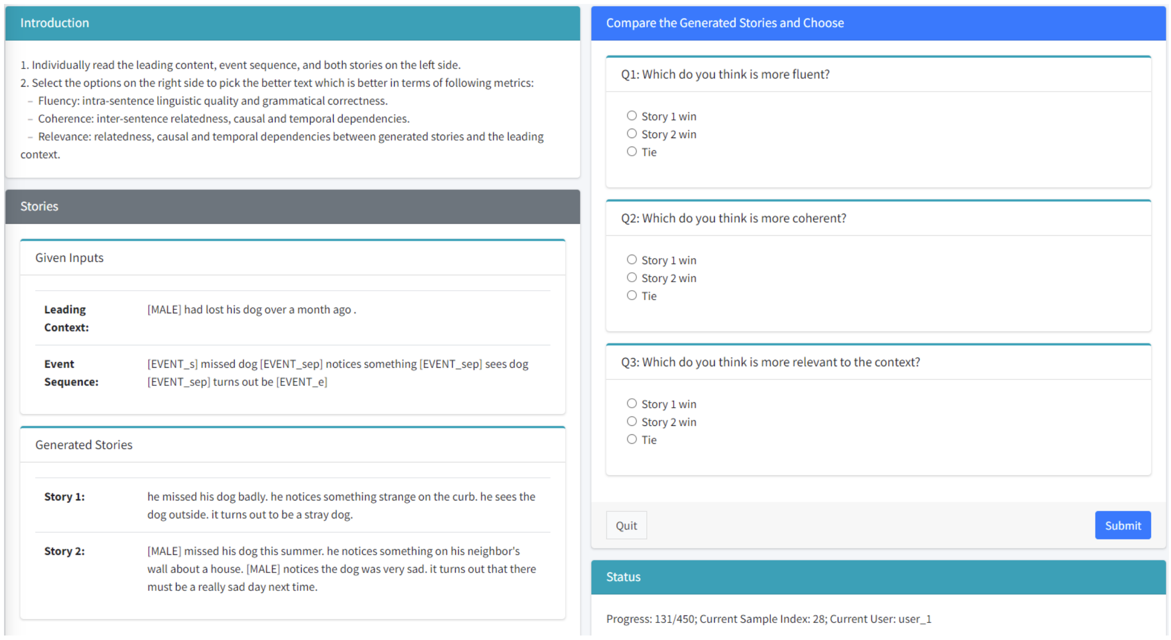
With temporally related events from training stories, we construct an event graph denoted as G, which is an isomorphic graph with single event type and single relation type. We suppose G is a data structure composed by triples in e_h,r,e_t format. The workflow of the extraction process is explained as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

```
Algorithm 1: Extract Event Sequence $E$
  Input: A story $S$ with $m$ sentences
  Output: Event Sequence
          $E$ for $S$ containing $m$ event objects
1 Initialise $E \leftarrow \varnothing$
   and $roles \leftarrow \{trigger, mod, agent, comp\}$
   foreach $s^i$ in $S$ do
2     Initialise $e_i \leftarrow \varnothing$
3     Normalise $s^i$
       and get dependencies $dep_i$ with $spaCy$
4     Extract
       event trigger $t$ and position $p_t$ from $dep_i$
5     $e_i.trigger \leftarrow t$
6     foreach $role$ in $role$ do
7         if $t \in dep_i.heads$
           and $role \in dep_i.tails$ then
8             Extract $(role, p_r)$ from $dep_i$
9             $e_i.role \leftarrow (role, p_r)$
10    $e_i.string \leftarrow r \in roles$ aligned by $p_r \uparrow$
11    $E$ append $e_i$
```

A.4 Details of Methodology

Sentence Prediction Task An auto-regressive decoder will predict y_t based on prior tokens $y_{<t}$, so we let a neural model learn to generate a special hidden state H^i_{sep} at the position of special token $[sep_i]$ where i denotes the i-th sentence. We use

[11]https://spacy.io/



Introduction

1. Individually read the leading content, event sequence, and both stories on the left side.
2. Select the options on the right side to pick the better text which is better in terms of following metrics:
  - Fluency: intra-sentence linguistic quality and grammatical correctness.
  - Coherence: inter-sentence relatedness, causal and temporal dependencies.
  - Relevance: relatedness, causal and temporal dependencies between generated stories and the leading context.

Stories

Given Inputs

Leading Context:	[MALE] had lost his dog over a month ago .
Event Sequence:	[EVENT_s] missed dog [EVENT_sep] notices something [EVENT_sep] sees dog [EVENT_sep] turns out be [EVENT_e]

Generated Stories

Story 1:	he missed his dog badly. he notices something strange on the curb. he sees the dog outside. it turns out to be a stray dog.
Story 2:	[MALE] missed his dog this summer. he notices something on his neighbor's wall about a house. [MALE] notices the dog was very sad. it turns out that there must be a really sad day next time.

Compare the Generated Stories and Choose

Q1: Which do you think is more fluent?

○ Story 1 win
○ Story 2 win
○ Tie

Q2: Which do you think is more coherent?

○ Story 1 win
○ Story 2 win
○ Tie

Q3: Which do you think is more relevant to the context?

○ Story 1 win
○ Story 2 win
○ Tie

Status

Progress: 131/450; Current Sample Index: 28; Current User: user_1

Figure 8: The snapshot of our evaluation interface. The stories are randomly collected from the **ROC** dataset, and the annotators are required to choose a choice for each question on the right colomn. For the convenience of accurate annotation, the system allow annotators to directly compare different generated stories with given input. A survey will be automatically recorded with all the three questions answered and the "submit" button pressed.

Sentence-Bert to obtain a numeric vector F_i^{sent}, which contains the features of sentences through representation learning. Then we force the similarity score sim_{ij}^s between generated sentences to fit similarities between sentences in the reference stories. The calculation of similarity score is shown below:

$$F_i^{sent} = \textit{Sentence-Bert}(\{s_1^i, \ldots, s_n^i\}) \tag{13}$$

$$sim_{ij}^s = cosine(F_i^{sent}, F_j^{sent}) \tag{14}$$

$$u_{ij} = (H_{sep}^i)^\intercal W^{sep} H_{sep}^j \tag{15}$$

$$sim_{ij}^y = sigmoid(u_{ij} + u_{ji}) \tag{16}$$

where i and j denote the indices of a sentence. sim denotes the similarity. sim_{ij}^s, the ground-truth similarity, is computed by $cosine$ similarity between outputs of *Sentence-Bert*. u_{ij} is an intermediate variable of similarity obtained from predicted sentence representations, and W^{sep} denotes a trainable parameter. To guarantee sim_{ij}^y is symmetric with respect to either i to j or j to i, u_{ij} and u_{ji} are both incorporated.

Contextualised Features Representation Conventional end-to-end models usually concatenate embeddings of different input, e.g. here we concatenate C and E as the input to the baseline model. The encoders such as LSTM, Bert-like encoders will incorporate the heterogeneous features. However, there may be some potential problems: (i) C and E have different word distribution, since C is natural language but E is not. A single encoder may not capture the two type of features in a single input numeric vector, efficiently and effectively; (ii) With the increment of the length of stories, the relative size of E will surpass C. The single encoder may pay less attention to C features, e.g. the vectors C involved in the calculations of attention scores become relatively less when the plan events increase.

A.5 Details of Human Evaluation

We developed an evaluation system, which helps us to collect annotations of evaluation, to make the story pairs for annotation anonymous, fairly shuffled, and easy to compare. Figure 8 is the snapshot of our annotating process.

Evaluators are required to follow the annotation standards shown on the left top corner. Considering the different biases among individuals, we also notify every annotator our standards set for this task: (i) **Fluency** considers the errors shown in generated text, e.g. grammatical errors $\geq$ spelling errors $\geq$ unnatural repetitions >language quality. (ii) **Coherence** focuses on the logical relatedness between sentences. We asked annotators to count all the incoherent parts, and consider how many word edits would be needed to make the story coherent (i.e. fewer edits needed = more coherent story). (iii) **Relevance** focuses on the relatedness between generated sentences and

Input	**Leading Context** [MALE] had lost his dog over a month ago . **Event Sequence** missed dog → notices something → sees dog → turns out be
P&W$_{l+e}$	He wished he could live with his friend. He 'd run in them all the time. But one day, he woke up exhausted. He went to the doctor with his best friend.
GPT-2$_{l+e}$	[MALE] was only a parent at the time. [MALE] **notices** the dog and he lets it go. He notices the dog has been moved and so he notices what happened. [MALE] then realises that it is a bad dog and there is something wrong with his life.
BART$_{l+e}$	He **missed** his **dog** for a whole month. One day he **notices something** moving and is startled. He **sees** the **dog** on the floor. It **turns out** to **be** a squirrel.
HINT$_{l+e}$	One day [MALE] **missed** his **dog**. He **notices something** about her name on the dog's tag. [MALE] **sees** the **dog** in the tags. It **turns out** it could **be** a dog from the police department.
EtriCA	He **missed** his **dog** badly. He **notices something** strange on the curb. He **sees** the **dog** outside. It **turns out** to **be** a stray dog.
- w/o sen	He had **missed** his **dog** so much that he had to search for him. As he was searching, he **notices something** about a dog. He **sees** the **dog** with a bag. It **turns out** to **be** a stray, a wad of dog spray.
- w/o cm	[MALE] **missed** his **dog** this summer. He **notices something** on his neighbor's wall about a house. [MALE] notices the dog was very sad. it **turns out** that there must **be** a really sad day next time.
- w/o leading	He **missed** his **dog**. [MALE] **notices something** in the area. he **sees** a **dog**. it **turns out** to **be** a black dog.
- w/o events	He was devastated by the loss. He decided to pull a long string of nail polish. He found a couple of old nail polish cans that were very old. His dog enjoyed his touches.

Table 7: A case study of generated stories conditioned with a leading context and an event sequence collected from **ROC Stories**. *[MALE]*, *[FEMALE]*, and *[NEUTRAL]* are the specital tokens to replace names in story. The highlighted bold words denote the events corresponding to the given event sequence.

the leading context. However, it is very subjective to judge if a story is "interesting" or not relevant. Therefore, we suggest evaluators to judge how irrelevant a story is by counting the conflict generated sentences with the leading context.

A.6 Case Study

As is shown in Table A.6, EtriCA can generate better stories considering both context relatedness and story quality. The strong baselines models, i.e. **BART**$_{l+e}$ and **HINT**$_{l+e}$, generate stories with good obedience to the planned event sequences, and relatively good fluency. However, they fail to write reasonable stories with the logical relatedness to the ongoing circumstances. For instance, "It turns out to be a squirrel." is a good sentence and also uses the event "turns out be", but it has nothing to do with the topic - "the dog is missing", and no coherence with previous sentences as well.

In terms of the results obtained for the ablation study, we see the importance of different components to the whole generation model. If there is no planned event sequence (see **- w/o events**), it is very hard to let a neural model write a coherent story, which is also demonstrated in previous work (Yao et al., 2019). If there is no given leading context (see **- w/o leading**), neural models struggle to unfold the planned events, because the neural model does not understand the concept of a "topic" for a story, it may cause confusion. Without the contextualising module (see **- w/o cm**), the neural model struggles to process the heterogeneous features from context and events.