Enhancing Text Generation with Inductive Event Reasoning

Anonymous ACL submission

Abstract

How to generate informative, coherent natural language is a very important task. Previous studies mainly focus on leveraging commonsense knowledge into generative models, which can improve the informativeness of generated texts. However, these models pay little attention to discourse coherence. Instead, we propose to utilize event chains to improve the coherence of text generation. In addition, we devise an inductive encoding module to reduce the sparsity of introduced event chains and learn the useful event evolution patterns. Specifically, we first extract event chains for the input text and then connect them as a graph. The inductive graph encoding module is then used to learn the inductive and generalized event embeddings. The event reasoning flow module follows and produces the event sketch, i.e., the reasonable events conditioned by the input text. Finally, we generate the text based on the input context and the event sketch. Experimental results indicate the effectiveness of this framework in terms of coherence and informativeness of text generation.

1 Introduction

Text generation aims to produce realistic and reasonable textual content that is indistinguishable from human-written text, and is helpful for generative question answering (Yin et al., 2015), neural conversation (Li et al., 2016) systems, etc.

To this end, end-to-end generative models have been studied (Shang et al., 2015; Shao et al., 2017), but these models tend to produce generic texts. Recently, some works propose to generate natural language by grounding on external knowledge (Zhou et al., 2018; Zhang et al., 2019; Ji et al., 2020). However, these methods tend to produce less coherent texts, because they focus on improving the informativeness of the generated texts, and pay little attention to the coherence to context scenarios (Xu et al., 2020).

In this paper, we take a step towards informative and coherent natural language generation. To achieve this goal, we propose to use the knowledge of narrative event chains. A narrative event chain is a partially ordered set of events centered around a common protagonist (Chambers and Jurafsky, 2008). Because event chains contains rich temporal and causal information (Zhao et al., 2017; Sap et al., 2019), previous study shows that the use of event chains as background knowledge leads to better coherence judgement of real narrative instances in a narrative cloze task (Chambers and Jurafsky, 2008; Li et al., 2018). Motivated by this fact, We use event graphs which are composed of event chains to enhance text generation since the graphs might help the event-based content planning (Xu et al., 2020), and conditioned on the graphs makes it easier to generate coherent texts. Figure 1 provides a case of using a event graph in αNLG.

However, using event graphs as knowledge grounding is non-trivial and two challenges exist. First, an observed event is very likely to appear only once, which brings about huge difficulty to learn frequent event evolution patterns. In fact, the in-degree of most event nodes (accounting for about 91%) in the event graphs is 1, which shows that the event graphs are extremely sparse. Second, due to the one-to-many relations between the context and the event knowledge, not all events are good to support coherent text generation. For example, the event "get ticket" and "buy food" in Figure 1 are not good ones. The model should be able to learn a more reasonable structure for capturing useful event patterns and reducing the impact of irrelevant event knowledge.

To address the aforementioned challenges, we propose a novel *Inductive Event Reasoning* (termed as IER) based method for text generation. Specifically, we devise an inductive event graph encoding module (§ 3.3.2) to learn inductive event embeddings, hence reducing the sparsity of event graphs

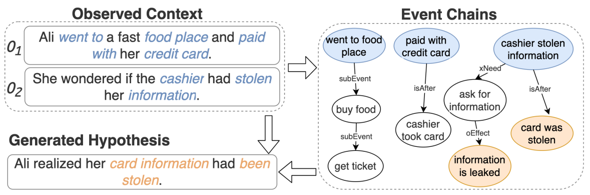


Figure 1: A case of using a event graph in *Abductive* NLG (αNLG). The αNLG aims to generate an explanatory hypothesis given two observations: O_1 as the cause and O_2 as the consequence. Blue nodes correspond to events in the context, orange nodes correspond to events used to generate the hypothesis.

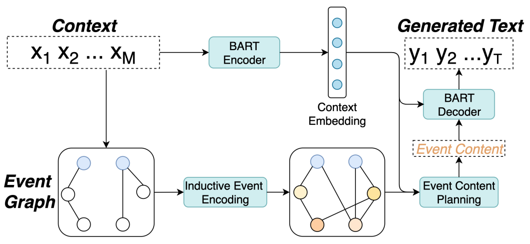


Figure 2: The workflow of our method. Blue nodes correspond to events in the context.

and learning useful event patterns. Then we make event-based content planning (§ 3.3.3) upon the learned event embeddings. We regard the planned events to be the skeleton of the expected scenario, so we further rewrite the planned events into natural language texts (§ 3.3.4). Experimental results on three widely used datasets demonstrate that our model substantially outperforms previous state-of-the-art methods.

Our contributions are summarized as follows: 1) We propose a event-based reasoning framework for text generation. It first makes a content-plan upon the event knowledge and then generate coherent texts. 2) We utilize inductive graph encoding module to reduce the sparsity of events and learn useful event evolution patterns, hence makes a more accurate event-based content planning. 3) We conduct extensive experiments, including automatic and manual evaluations, on three deeply studied text generation tasks. Our study indicates that both the knowledge of event chains and our reasoning modules are crucial to our superior performance in coherence of text generation.

2 Related Work

2.1 Knowledge-Aware Text Generation

Sequence-to-sequence models, e.g., (Sutskever et al., 2014), have been widely used for natural language generation. Earlier works focus on utilizing external knowledge to augment the limited textual information. (Zhou et al., 2018) enriched the representations of the input texts with neighbouring concepts on ConceptNet using graph attention. (Guan et al., 2019) proposed incremental encoding with multi-source attention to incorporate one-hop knowledge graph for concepts in the story context. (Zhang et al., 2019) models the conversation flow explicitly with the commonsense knowledge graph and guides the conversation flow in the latent concept space. Nevertheless, the degenerating of irrelevant, off-topic, and non-useful generated texts is still one of the main challenges. Recent works attempt to integrate external knowledge into pretrained language models to generate more informative texts. (Guan et al., 2020) conducted post-training on synthetic data constructed from commonsense knowledge bases by translating triplets into natural language texts using templates. (Ji et al., 2020) performs multi-hop reasoning on the external knowledge graph for knowledge-enriched language generation. Different from those work, we introduce the knowledge of event chains to generate more coherent texts, which is less studied in previous works.

2.2 Text Generation with Content Planning

Text generation with content planning first identifies pertinent information to present, and then realizes the pertinent information into surface text (Holmes-Higgin, 1994). Traditional systems often handle each step separately, thus requiring extensive effort on data acquisition and system engineering (Reiter and Dale, 1997). Recent progress has been made by developing end-to-end trained neural models (Yu et al., 2018; Fan et al., 2018). Nonetheless, the incoherent and unfaithful generations is still the limitation of these models. In this paper, we make the content planning based on event chains, i.e., perform event graph grounded reasoning flow, to generate texts. The event chains contain rich temporal and causal information (Zhao et al., 2017; Sap et al., 2019), which guarantees the coherence of generated texts.

3 Methodology

3.1 Task definition

The input $X = (x_1, x_2, \cdots, x_M)$ is a text sequence which may consist of several sentences. The output target is another text sequence $Y =$

$(y_1, y_2, \cdots, y_T)$. To facilitate the generation process, according to X, we extract a event graph $G = \{\mathcal{V}, \mathcal{E}\}$ as knowledge grounding, where $\mathcal{V}$ denotes the event set and $\mathcal{E}$ denotes the relations connecting these events. Given the input sequence X and the event graph G, we decompose the text generation task into two steps. The first step makes the event-based content planning to select the most reasonable k events E_k from G according to X. In the second step, we rewrite E_k into Y conditioned on the context X. Essentially, the model maximizes the following conditional probability:

$$P(Y|X,G) = P(E_k|X,G) \cdot P(Y|X,E_k). \quad (1)$$

The framework of our method is shown in Figure 2. Next, we introduce the event graph construction.

3.2 Constructing Event Graph

To obtain event knowledge for language generation, we use ATOMIC (Sap et al., 2019) and COMeT (Hwang et al., 2020) as event knowledge base. ATOMIC is a repository of inferential if-then knowledge. COMeT is a transformer model trained on ATOMIC that generates nine kinds of inferences of events in natural language.[1]

Given the input context X, we first extract central events from X, as described in Figure 1. The central events are feed into COMeT model to generate one-hop events with corresponding relations. The one-hop events are then feed into COMeT to generate two-hop events. As a result, we obtain lots of event chains. We also apply several heuristic rules to filter less informative chains, as introduced in § 4.2. Then, we merge the retained event chains into a multi-hop event graph. Figure 1 provides a example of the constructed event graph.

3.3 Generation through Inductive Event Reasoning

3.3.1 Context Encoding

Given the context $X = (x_1, x_2, \cdots, x_M)$, we use pretrained BART (Lewis et al., 2019) encode to encoder each token x_m into the hidden state $\mathbf{h}_m \in \mathcal{R}^d$ (d is the hidden size) with self-attention mechanism (Vaswani et al., 2017):

$$\{\mathbf{h}_m\}_{m=1,\cdots,M} = \text{BARTEncoder}(X). \quad (2)$$

Then we obtain the hidden state $\mathbf{h}_X \in \mathcal{R}^d$ of X by

$$\mathbf{h}_X = \mathbf{W}_x[\text{max}_m(\mathbf{h}_m); \text{ave}_m(\mathbf{h}_m)], \quad (3)$$

[1]The relations used in our work includes: *Causes*, *xIntent*, *xNeed*, *oEffect*, *oWant*, *isAfter*, *HasSubEvent*.

where max$(\cdot)$, ave$(\cdot)$ denotes the max-pooling and ave-pooling respectively, $[\cdot;\cdot]$ denotes the concatenation, $\mathbf{W}_x$ is a trainable. For simplicity, we omit the details of BART, and refer the readers to the original work (Lewis et al., 2019).

3.3.2 Inductive Encoding of Event Graph

To capture useful event knowledge and reduce the impact of irrelevant events in the event graph, our model treats the reasoning structure as a latent variable and induces it with the input event graph. We call the induced structure as Induced Graph (G_I). The induction module is built based on the structured attention (Kim et al., 2017). We use a variant of Kirchhoff's Matrix-Tree Theorem (Koo et al., 2007; Nan et al., 2020; Cao et al., 2021) to induce the latent structure.

Formally, the nodes of G_I are the events in G, and suppose the number of nodes is N. Let $\mathbf{e}_i \in \mathcal{R}^d$ denotes the initialized embedding of i-th node in the event graph G, and $\mathbf{e}_i$ can be obtained via the pretrained model (i.e., BART). We first calculate the pair-wise unnormalized attention score s_{ij} between the i-th node and the j-th node:

$$\mathbf{s}_{ij} = (\sigma(\mathbf{W}_p\mathbf{e}_i))^T\mathbf{W}_b(\sigma(\mathbf{W}_c\mathbf{e}_j)), \quad (4)$$

where $\mathbf{W}_p$, $\mathbf{W}_b$ and $\mathbf{W}_c$ are trainable, σ is the *tanh* activation. Next, we compute the root score $\mathbf{s}_i^r$ which represents the unnormalized probaility of the i-th node to be selected as the root node of G_I:

$$\mathbf{s}_i^r = \mathbf{W}_r\mathbf{e}_i, \quad (5)$$

where $\mathbf{W}_r \in \mathcal{R}^{1\times d}$ is trainable.

Following (Nan et al., 2020), we calculate the marginal probability of each edge of G_I. We first assign non-negative weights $\mathbf{P} \in \mathcal{R}^{N\times N}$ to the edges of the induced graph:

$$\mathbf{P}_{ij} = \begin{cases} 0, & \text{if } i = j \\ \exp(\mathbf{s}_{ij}), & \text{otherwise,} \end{cases} \quad (6)$$

where $\mathbf{P}_{ij}$ is the weight of the edge between the i-th and j-th node. We then define the Laplacian matrix $\mathbf{L} \in \mathcal{R}^{N\times N}$ of G_I, and its variant $\hat{\mathbf{L}} \in \mathcal{R}^{N\times N}$ for further computations (Koo et al., 2007):

$$\mathbf{L}_{ij} = \begin{cases} \sum_{k=1}^{N} \mathbf{P}_{kj}, & \text{if } i = j \\ -\mathbf{P}_{ij}, & \text{otherwise,} \end{cases} \quad (7)$$

$$\hat{\mathbf{L}}_{ij} = \begin{cases} \exp(\mathbf{s}_i^r), & \text{if } i = 1 \\ \mathbf{L}_{ij}, & \text{otherwise.} \end{cases} \quad (8)$$

We use $\mathbf{A}_{ij}$ to denote the marginal probability of the edge between the i-th node and the j-th node, which can be computed as follows:

$$\mathbf{A}_{ij} = (1-\theta_{1,j})\mathbf{P}_{ij}[\hat{\mathbf{L}}^{-1}]_{ij} - (1-\theta_{i,1})\mathbf{P}_{ij}[\hat{\mathbf{L}}^{-1}]_{ji}, \quad (9)$$

where θ is the Kronecker delta (Koo et al., 2007) and $\cdot^{-1}$ denotes matrix inversion. $\mathbf{A}$ can be regarded as a weighted adjacency matrix of G_I, and is used for induced structure-aware event encoding.

To better capture potential reasoning clues, we adopt the densely connected graph convolutional networks (DCGCNs) (Guo et al., 2019), which allows training a deeper reasoning model. The convolution computation of each layer is:

$$\mathbf{h}_i^{(l)} = \sigma(\sum_{j=1}^{N} \mathbf{A}_{ij}\mathbf{W}_v^{(l)}\mathbf{h}_j^{(l)} + \mathbf{b}_v^{(l)}), \qquad (10)$$

where $\mathbf{W}_v^{(l)}$, $\mathbf{b}_v^{(l)}$ are parameters, σ is the ReLU activation, $\mathbf{h}_j^{(0)}$ is the initial event embeddings.

The induced structure at once is relatively shallow (Nan et al., 2020; Cao et al., 2021) and may not be optimal for event-based reasoning. Therefore we iteratively refine the induced structure to learn a more informative one. We stack N blocks of this module, where each block contains a induced structure and a DCGCNs network. After the iterative refinement, the induced event embeddings are used for event-based content planning.

3.3.3 Event Graph Based Content Planning

Given the context X and the induced event embeddings, we perform the event graph based content planning to produce the event sketch, i.e., the most reasonable events conditioned the input X. Due to the temporal and causal information carried by the event graph, the produced event sketch improves the coherence of text generation.

Inspired by the multi-hop reasoning flow (Ji et al., 2020), the module iteratively computes the scores between X and multi-hop events. For the candidate event $t_e \in \mathcal{V}$, the module computes the attention score $s(t_e)$ between t_e and X by aggregating evidence from its neighbours $\mathcal{N}_{t_e}$ including pairs of the connected event $u_e \in \mathcal{V}$ and the connected edge $r_e \in \mathcal{E}$:

$$s(t_e) = \frac{1}{|\mathcal{N}_{t_e}|}\sum_{(u_e,r_e)\in\mathcal{N}_{t_e}}(\gamma \cdot cs(u_e) + R(u_e,r_e,t_e)), \quad (11)$$

where γ (0.5 by default) is a discount factor that controls the attention flow from the previous hops. Initially, the zero-hop events, which are extracted from X, are given a score of 1, while other events are assigned with 0. $R(\cdot)$ is the relevance of the triple (u_e, r_e, t_e) under the context $\mathbf{h}_X$, which is calculated by:

$$R(u_e,r_e,t_e) = \sigma(\mathbf{h}_X^\top \mathbf{W}_s[\mathbf{h}_{u_e};\mathbf{h}_{r_e};\mathbf{h}_{t_e}]), \quad (12)$$

where $\mathbf{W}_s$ is trainable, σ denotes *sigmoid* activation, $[\cdot;\cdot]$ denotes the concatenation, $\mathbf{h}_{u_e}$ and $\mathbf{h}_{t_e}$ are induced embeddings (§ 3.3.2) of the event u_e and t_e, $\mathbf{h}_{r_e}$ is the relation embedding.

According to the attention scores (Eq. 11) of all events, we select the k events $E_k = \text{topk}_i(s(e_i))$, which are used as sketch for text generation. k is set to 2 by default.

3.3.4 Text Generation with Planned Events

The event sketch E_k describes the skeleton of the expected scenario, and can be used to guide text generation. Specifically, we first merge X and E_k to obtained the new context $C = [X; E_k]$, and use BART model to obtain the context vectors

$$\mathbf{H}_C = \text{BARTEncoder}(C), \qquad (13)$$

where $\mathbf{H}_C \in \mathcal{R}^{c\times d}$ (c is the total length of C). The we perform text generation with the BART model.

The hidden state of t-th time step of the target sequence $\mathbf{h}_{y_t}$ is computed by:

$$\mathbf{h}_{y_t} = \text{BARTDecoder}(\mathbf{Y}_{\le t}, \mathbf{H}_C), \qquad (14)$$

with $\mathbf{H}_C$ as the attended context in BARTDecoder (the details of BARTDecoder can be referred at (Lewis et al., 2019)). The word distribution of t-th time step over the standard vocabulary V is

$$P(y_t|Y_{<t}) = \text{softmax}_V(\mathbf{W}_v\mathbf{h}_{y_t} + \mathbf{b}). \qquad (15)$$

3.4 Training and Inference

To train the event planning module, we minimize the cross-entropy loss of selecting supported events under the context X by:

$$\begin{aligned} J_P = \frac{1}{Z}\sum_i\big(& -l_i \cdot log(p(e_i)) \\ & - (1-l_i)\cdot log(1-p(e_i))\big), \end{aligned} \qquad (16)$$

where $p(e_i) = \text{sigmoid}(s(e_i))$ denotes the probability that the event e_i is positive, l_i is the label of e_i, Z is the number of events in G. The details of event labeling is shown in § 4.2.

To train the text generator, we minimize the NLL loss of generating the ground-truth:

$$J_{\text{NLL}} = \sum_{t=1}^{T} -\log P(y_t^{\text{gold}} | Y_{<t}^{\text{gold}}). \tag{17}$$

The final loss is $J = J_P + J_{\text{NLL}}$. The event planner and the text generator are jointly training.

4 Experiments

4.1 Datasets

Story Ending Generation (SEG) is to generate a reasonable ending given a four-sentence story context. The stories come from ROCStories corpus (Mostafazadeh et al., 2016). We use the same data split as (Yao et al., 2019).
Abductive NLG (αNLG) is to generate an explanatory hypothesis given two observations: O_1 as the cause and O_2 as the consequence. We use the official data split from (Bhagavatula et al., 2019).
Explanation Generation (EG) is to generate an explanation given a counter-factual statement for sense-making (Wang et al., 2019). We use the same data split as (Ji et al., 2020).
The statistics of the datasets are shown in Appendix A, Table 6.

4.2 Event Chains Filtering

Given the context X, lots of event chains are generated, so we devise several heuristic rules to filter less informative chains. For example, if an event in a chain contains less than 2 words, the chain will be discarded. There are still many event chains, which may lead to a expensive memory cost, so we reserve 80 event chains according to the length of terminal events on the chain. Then, we merge the retained event chains into an event graph. We heuristically label events according to the word overlap between an event and the corresponding ground-truth. The word overlap is calculated by the Longest Common Substring (LCS) algorithm. The statistics of extracted event graphs is shown in Appendix A, Table 7.

4.3 Baselines

We produce the following baselines on three generation tasks to compare with our model:
GPT2-FT is a GPT-2 model fine-tuned on the task-specific dataset with its model initialization from (Radford et al., 2019). **BART-FT** is a BART model fine-tuned on the task-specific dataset with its model initialization from (Lewis et al., 2019).
GPT2-OMCS is a commonsense-enhanced GPT-2 model first post-trained on the Open Mind Common Sense (OMCS) corpus5 from which the ConceptNet (Speer et al., 2017) is constructed. The model is then fine-tuned on the task-specific dataset.
GRF-GPT2 (Ji et al., 2020), the current state-of-the-art model on the used datasets, is a GPT2 based model which generates natural language texts with multihop reasoning on knowledge graph. For the sake of fairness, we also use the BART pretrained model to reproduce GRF and name it **GRF-BART**.

We also compare our model with baseline models designated to each specific task. For story ending generation, we compare to **WriterForcing** (Gupta et al., 2019) that forces the attention to focus on important key-phrases and avoid generating generic words. For abductive NLG, we compare with **COMeT-Txt** (Bhagavatula et al., 2019) which uses the output texts generated by COMeT (Hwang et al., 2020) as prefix texts to the GPT2 model while fine-tuning.

4.4 Implementation Details

Our method employs the base version of the BART (Lewis et al., 2019) model with 6 layers, 768-dimensional hidden states, and 12 attention heads for contextual modeling. Each refinement block contains a two-layers DCGCNs module. To train the model, we use the Adam optimizer (Kingma and Ba, 2014) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-6}$ and linearly decrease learning rate to zero with no warmup. We search for the best hyper-parameters according to BLEU-2 on the development set of each dataset. At the inference stage, we adopt beam search decoding with a beam size of 3 for our model and all the baselines we produce.

4.5 Main Results

4.5.1 Automatic Evaluation

Metrics: For automatic evaluation, we use metrics including BLEU-4 (Papineni et al., 2002), ROUGE-L (Lin, 2004), METEOR (Banerjee and Lavie, 2005) and CIDEr (Vedantam et al., 2015) to evaluate the αNLG and the EG tasks. We use BLEU-1/2 and METEOR to evaluate the SEG task.
Result: Automatic evaluation results on the test sets of three tasks are shown in Table 1 and 2. Our implementation of GRF (GRF-BART) performs better than the original one (Ji et al., 2020), the improvement is caused by the BART model. In ad-

Models	αNLG				EG			
	BLEU-4	ROUGE-L	METEOR	CIDEr	BLEU-4	ROUGE-L	METEOR	CIDEr
COMeT-Txt	2.73	24.39	18.32	32.78	N/A	N/A	N/A	N/A
GPT2-FT	9.80	32.90	25.82	57.52	15.63	37.32	38.76	77.09
BART-FT	12.46	33.74	28.81	66.77	16.48	37.08	41.25	81.56
GPT2-OMCS	9.62	32.88	25.83	57.50	15.55	37.53	38.28	75.60
GRF-GPT2*	11.62	34.62	27.76	63.76	17.19	38.10	39.15	81.71
GRF-BART†	13.47	35.53	30.82	72.01	17.74	38.06	41.47	87.35
IER (*Ours*)	**16.05**	**37.20**	**33.14**	**77.87**	**18.91**	**38.74**	**42.97**	**91.74**

Table 1: Automatic evaluation results on the testsets of αNLG and EG. Entries with N/A mean the baseline is not designated for this task. * denotes we use the generated file from (Ji et al., 2020). † means our implementation based on BART (Lewis et al., 2019) pretrained model.

Models	BLEU-1/2	ROUGE-L	METEOR
WriterForcing	16.5/3.7	N/A	N/A
GPT2-FT	25.5/10.2	N/A	N/A
BART-FT	25.6/10.4	26.26	18.75
GPT2-OMCS	25.5/10.4	N/A	N/A
GRF-GPT2*	26.1/11.0	26.62	19.36
GRF-BART†	26.1/11.2	26.91	19.64
IER (*Ours*)	**27.2/12.7**	**28.75**	**21.38**

Table 2: Automatic evaluation on the test set of SEG. Entries with N/A denotes the value is not reported. * means we use the generated file from (Ji et al., 2020). † denotes our implementation based on BART model.

dition, our model outperforms all baselines that utilize commonsense knowledge or pretrained models. We have following observations: **First**, our method significantly outperforms COMeT-Txt, which also uses event knowledge from COMeT. We think this is because we perform reasoning flow on event graphs and use the inductive encoding module to learn useful event relations. **Second**, compared with GRF-BART, our model achieves a notably improvement. This indicates that the knowledge of event chains is especially important for improving the coherence of text generation. **Third**, our model performs better on the αNLG and SEG than it on EG. After analysis, we find that αNLG and SEG emphasize the discourse relations between narrative texts, therefore, the event chains contribute more to the two datasets.

4.5.2 Manual Evaluation

Metrics: For manual evaluation, all models are evaluated in terms of the following 2 metrics: informativeness, coherence. For informativeness, the annotators are required to assess whether a generated text produces unique and non-genetic information that is specific to the input context. When evaluating coherence, for SEG and αNLG, annotators are asked to focus on evaluating the causal and temporal relevance of a generated text and the context; for the EG, annotators are mainly asked to check whether the generated text explains the counterfactual points in the statement properly.

Result: We carried out pairwise comparison with BART-FT, GRF-GPT2, and GRF-BART on the used three datasets. We randomly sample 100 sentences from the three test sets respectively for each pair of models. We recruit two annotators to make a preference among win, tie and lose given the input context and two outputs generated by our model and a baseline respectively. The annotators are both graduates from the field of text generation. As shown in Table 3, our model significantly outperforms compared baselines in terms of both criteria on all the datasets. Both automatic and manual evaluations illustrate the effectiveness of our method for text generation. The inter-rater agreement is shown in Appendix B, Table 8.

4.5.3 Ablation Study

To investigate the effectiveness of different modules, we develop three ablated versions, including (1) “w/o IE” which denotes we ablate the inductive encoding module of event graph and directively performs event planning on the intialized event embeddings, (2) “w/ RGCNs” which denotes we replace the IE module with Relational-GCNs (RGCNs) (Schlichtkrull et al., 2018) to learn structure-aware event embeddings, (3) “w/o J_P” which denotes we *do not train* the event planning module.

Models	αNLG				SEG				EG			
	Informativeness		Coherence		Informativeness		Coherence		Informativeness		Coherence	
	W	L	W	L	W	L	W	L	W	L	W	L
vs. BART-FT	51.5	8.0	46.5	3.0	30.5	1.5	31.0	1.5	21.5	1.5	26.5	2.5
vs. GRF-GPT2	40.5	7.5	47.0	4.5	30.0	2.0	32.0	5.0	23.5	4.0	31.5	5.5
vs. GRF-BART	37.0	7.0	39.5	9.0	21.0	1.0	20.0	4.0	18.0	2.5	23.0	6.5

Table 3: The manual evaluation results on the three testsets. Scores indicate the percentage (%) of Win (W) and Lose (L) when comparing our model with a baseline in terms of two metrics.

Models	αNLG				SEG			
	BLEU-4	ROUGE-L	METEOR	CIDEr	BLEU-1	BLEU-2	ROUGE-L	METEOR
Full	16.05	37.20	33.14	77.87	27.2	12.7	28.75	21.38
w/o IE	15.57	36.83	32.71	76.25	26.5	12.2	28.36	20.72
w/ RGCNs	15.78	36.80	32.81	76.61	26.6	12.3	28.51	20.94
w/o J_P	15.22	36.51	32.46	75.45	26.2	11.9	28.03	20.42

Table 4: Ablation study results on the testsets of αNLG and SEG. IE denotes the inductive encoding module of event graph (see § 3.3.2).

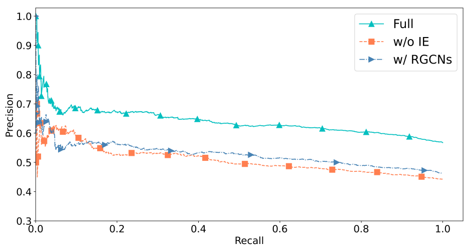


Figure 3: The Precision-Recall curves of the different models for event-based content planning at αNLG testset. The compared models include the full model and the two ablated versions: "w/o IE" and "w/ RGCNs".

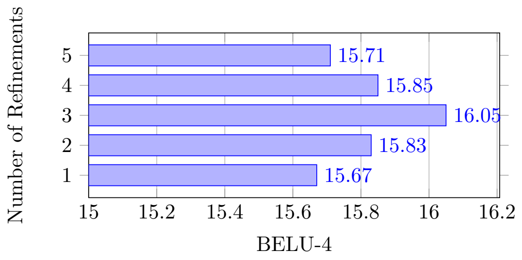


Figure 4: BLEU-4 for different number of refinements (i.e., N) on the αNLG dataset. The number of refinements is ranging from 1 to 5.

Table 4 shows the ablation results for text generation. We also evaluate their performance for event planning, on αNLG dataset. The Precision-Recall curves of the models are drawn and shown in Figure 3. The results in Figure 3 is consistent with the results in Table 4. We observe that: **First**, "w/o J_P" leads to a huge result drop. This coincides with human intuition because, with the help of supervision, the event planner learns to select useful event knowledge for text generation. **Second**, the result shows a notable decrease after ablating the IE module. This indicates the IE module learns the induced event embeddings and helps to reduce the irrelevant events when event planning. **Third**, although RGCNs has been proved to be effective in modeling the relational information in the knowledge graph, the "w/ RGCNs" version performs similarly to the "w/o IE" version (just a minor increase). This fact supports our argument that it is hard to select useful event knowledge from the sparse event graph.

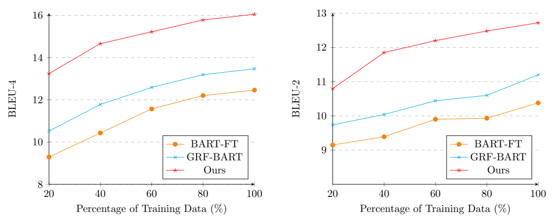
4.6 Model Analysis

4.6.1 Impact of the Number of Refinements

We investigate the effect of the refinement on the overall performance. We plot the BLEU-4 varying with the number of refinements in Figure 4. From the figure we can observe that:
(1) Our method yields the best performance in the third refinement. Compared with the first induction, the second refinement achieves 0.38 improvements of BLEU-4 on the αNLG dataset. This indicates that the proposed Inductive Encoding module is

able to induce more reasonable reasoning structures by iterative refinement. (2) When the number of refinements is too large, the performance decreases due to over-fitting.

4.6.2 Impact of the Size of Training Data



(a) αNLG (b) SEG

Figure 5: Performance with different amount of training data on the test set of αNLG and SEG respectively.

To demonstrate the complementary performance gain of utilizing event knowledge besides textual modeling, we sample different fractions of training data of αNLG and SEG, respectively, for training and evaluate on the original test set. We compare our method with BART-FT and GRF-BART. As shown in Figure 5, our model achieves consistent performance gains against the chosen baselines with different amount of training data, and our method can still manage to achieve comparable performance with best result of GRF-BART, even when given extremely small training data (at the x-axis point of 20%). This demonstrates the generalization ability of the proposed model with the aid of the knowledge of event chains.

4.6.3 Impact of the Number of Event Chains

Number	BLEU-4	METEOR	CIDEr
40	15.64	32.65	76.95
60	15.84	32.65	76.94
80	16.05	33.14	77.87
100	15.57	33.05	77.46

Table 5: Performance with different amount of event chains on the test set of αNLG.

We also investigate the influence of the number of event chains on our method, the result is shown in Table 5. As the number of events increases to 80, the model achieves the best results. This is because more supported events are exposed and selected to facilitate text generation. However, as the number increases further, noise events begin to overwhelm the supported events. It becomes harder to distinguish the supported events from noise events, resulting in a performance drop. In addition, we set the number of event chains to 80 and 40 for SEG and EG datasets respectively.

4.7 Visualization

We provide three cases with the generated texts of corresponding models, as shown in Appendix C, Table 9. We observe that: Baseline models only focus on the commonsense relations between entities, hence fails to generate coherent text. For example, in the first case, GRF-BART adopts the relations between "(*cat*, *animal*)" and "(*scaredy*, *shelter*)", and generates the incoherent text. Instead, by using the knowledge of event chains, our method generates more informative and more coherent sentences.

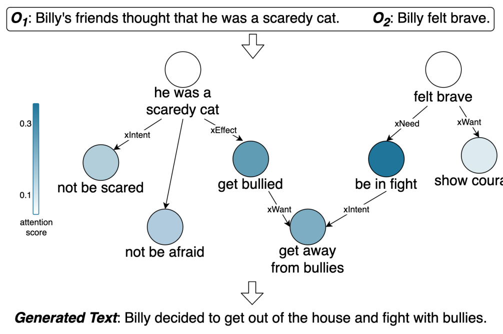


Figure 6: The local event subgraph of the first case, with the attentions scores of events.

To answer why our method generates satisfied sentences, we visualize the local event subgraph of the first case in Figure 6. We find that the event "*be in fight*" receives highest attention scores, then is the "*get bullied*", as a result, our method generate the coherent hypothesis.

5 Conclusion

We present a inductive event reasoning method for text generation. The proposed method uses the knowledge of event chains to improves the coherence of text generation. The inductive event encoding module is used to reduce the sparsity of events and learns the useful event evolution patterns. Extensive experiments show that our method outperforms existing state-of-the-art models. The ablation study demonstrates the effectiveness of different modules of our method. We also visualize our method with inferred event chains that provide a rationale for the generated results.

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Scott Wen-tau Yih, and Yejin Choi. 2019. Abductive commonsense reasoning. *arXiv preprint arXiv:1908.05739*.

Pengfei Cao, Xinyu Zuo, Yubo Chen, Kang Liu, Jun Zhao, Yuguang Chen, and Weihua Peng. 2021. Knowledge-enriched event causality identification via latent structure induction networks. 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 4862–4872.

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

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

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, Yansen Wang, and Minlie Huang. 2019. Story ending generation with incremental encoding and commonsense knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6473–6480.

Zhijiang Guo, Yan Zhang, Zhiyang Teng, and Wei Lu. 2019. Densely connected graph convolutional networks for graph-to-sequence learning. *Transactions of the Association for Computational Linguistics*, 7:297–312.

Prakhar Gupta, Vinayshekhar Bannihatti Kumar, Mukul Bhutani, and Alan W Black. 2019. Writerforcing: Generating more interesting story endings. *arXiv preprint arXiv:1907.08259*.

Paul Holmes-Higgin. 1994. Text generation—using discourse strategies and focus constraints to generate natural language text by kathleen r. mckeown, cambridge university press, 1992, pp 246,£ 13.95, isbn 0-521-43802-0. *The Knowledge Engineering Review*, 9(4):421–422.

Jena D Hwang, Chandra Bhagavatula, Ronan Le Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and Yejin Choi. 2020. Comet-atomic 2020: On symbolic and neural commonsense knowledge graphs. *arXiv preprint arXiv:2010.05953*.

Haozhe Ji, Pei Ke, Shaohan Huang, Furu Wei, Xiaoyan Zhu, and Minlie Huang. 2020. Language generation with multi-hop reasoning on commonsense knowledge graph. *arXiv preprint arXiv:2009.11692*.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M Rush. 2017. Structured attention networks. *arXiv preprint arXiv:1702.00887*.

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

Terry Koo, Amir Globerson, Xavier Carreras Pérez, and Michael Collins. 2007. Structured prediction models via the matrix-tree theorem. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 141–150.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Austin, Texas. Association for Computational Linguistics.

Zhongyang Li, Xiao Ding, and Ting Liu. 2018. Constructing narrative event evolutionary graph for script event prediction. *arXiv preprint arXiv:1805.05081*.

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.

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.

Guoshun Nan, Zhijiang Guo, Ivan Sekulić, and Wei Lu. 2020. Reasoning with latent structure refinement for document-level relation extraction. *arXiv preprint arXiv:2005.06312*.

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.

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.

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

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A Smith, and Yejin Choi. 2019. Atomic: An atlas of machine commonsense for if-then reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3027–3035.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1577–1586, Beijing, China. Association for Computational Linguistics.

Louis Shao, Stephan Gouws, Denny Britz, Anna Goldie, Brian Strope, and Ray Kurzweil. 2017. Generating high-quality and informative conversation responses with sequence-to-sequence models. *arXiv preprint arXiv:1701.03185*.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-first AAAI conference on artificial intelligence*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

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 *Advances in neural information processing systems*, pages 5998–6008.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Cunxiang Wang, Shuailong Liang, Yue Zhang, Xiaonan Li, and Tian Gao. 2019. Does it make sense? and why? a pilot study for sense making and explanation. *arXiv preprint arXiv:1906.00363*.

Jun Xu, Zeyang Lei, Haifeng Wang, Zheng-Yu Niu, Hua Wu, and Wanxiang Che. 2020. Enhancing dialog coherence with event graph grounded content planning. In *IJCAI*, pages 3941–3947.

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

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2015. Neural generative question answering. *arXiv preprint arXiv:1512.01337*.

Zhiwei Yu, Jiwei Tan, and Xiaojun Wan. 2018. A neural approach to pun generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1650–1660, Melbourne, Australia. Association for Computational Linguistics.

Houyu Zhang, Zhenghao Liu, Chenyan Xiong, and Zhiyuan Liu. 2019. Grounded conversation generation as guided traverses in commonsense knowledge graphs. *arXiv preprint arXiv:1911.02707*.

Sendong Zhao, Quan Wang, Sean Massung, Bing Qin, Ting Liu, Bin Wang, and ChengXiang Zhai. 2017. Constructing and embedding abstract event causality networks from text snippets. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 335–344.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018. Commonsense knowledge aware conversation generation with graph attention. In *IJCAI*, pages 4623–4629.

A Statistics of Datasets

The statistic for three datasets is shown in Table 6. The statistic of extracted event graphs for each dataset is shown in Table 7. *AveEventNum* denotes the average number of events in each event graph. *AveEdgeNum* denotes the average number of edges in each event graph. *PosEventGraph* denotes the ratio (%) of event graphs where each event graph contains at least one positive labeled event.

Tasks	Train	Dev	Test
SEG	78,526	9,818	9,818
αNLG*	50,481	7,252	14,313
EG*	25,596	1,428	2,976

Table 6: Statistics of the used datasets. *:Examples with multiple references are counted separately.

	αNLG	SEG	EG
AveEventNum	73.4	79.4	40.0
AveEdgeNum	78.2	82.5	43.7
PosEventGraph (%)	35.5	32.4	26.9

Table 7: The statistics of extract event graphs for three datasets.

B Inner-Rater Agreement

We calculate the cohen's kappa reliability as the inner-annotator agreement for manual evaluation. Scores in Table 8 evaluates the agreement from multiple annotators in terms of informativeness and coherence.

	αNLG	SEG	EG
Informativeness	0.732	0.598	0.888
Coherence	0.745	0.775	0.963

Table 8: The cohen's kappa reliability for manual evaluation.

C Case Study

Table 9 presents three cases where the first and second case is come from the αNLG testset, the third case comes from the SEG tesetset. Our method generate more informative and more coherent text sentences while baseline models tend to generate low quality sentences. Figure 7 illustrates the generate sentence of the second case.

#1	O_1	Billy's friends thought that he was a scaredy cat.
	O_2	Billy felt brave.
	BART-FT	Billy decided to go to the animal shelter.
	GRF-BART	Billy went to the animal shelter.
	Ours	Billy decided to get out of the house and fight with bullies.
#2	O_1	Frank was ready to go home after a long double shift.
	O_2	Frank was so grateful for Annie, as was there in his time of need.
	BART-FT	Frank's friend Annie came to help him.
	GRF-BART	Frank's friend Annie came over to help him.
	Ours	Frank got a call from Annie asking if he could get some rest.
#3	Story Context	Ryan stood at the counter with chocolate frosting outlining his mouth. He again denied eating the cupcake with a quick shake of his head. Her lightning-quick finger wiped across his face. The sugary brown evidence of his guilt frosted her finger .
	BART-FT	Ryan's mouth was swollen shut and he had to go to the bathroom.
	GRF-BART	Ryan was so embarrassed , he nearly fainted .
	Ours	Ryan's face turned red as he realized he'd forgotten the frosting.

Table 9: The three cases and the generated text of compared models. The first and second case come from the αNLG testset. The third case comes from the ROCStories testset.

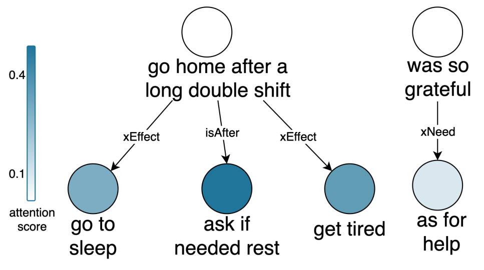


Figure 7: The local event subgraph of the second case.