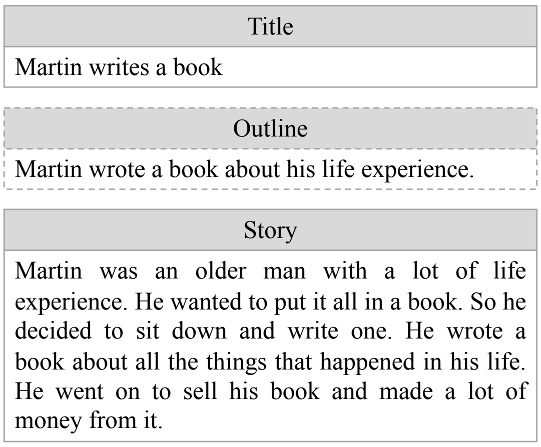
Learning to Predict Explainable Plots for Neural Story Generation

Gang Chen[†], **Yang Liu**[†‡], **Huanbo Luan**[†], **Meng Zhang**[#], **Qun Liu**[#] **and Maosong Sun**[†]
[†]Institute for Artificial Intelligence
State Key Laboratory of Intelligent Technology and Systems
Department of Computer Science and Technology, Tsinghua University, Beijing, China
[‡]Beijing National Research Center for Information Science and Technology
[#]Noah's Ark Lab Paris, Huawei Technologies Ltd

Abstract

Story generation is an important natural language processing task that aims to generate coherent stories automatically. While the use of neural networks has proven effective in improving story generation, how to learn to generate an explainable high-level plot still remains a major challenge. In this work, we propose a latent variable model for neural story generation. The model treats an outline, which is a natural language sentence explainable to humans, as a latent variable to represent a high-level plot that bridges the input and output. We adopt an external summarization model to guide the latent variable model to learn how to generate outlines from training data. Experiments show that our approach achieves significant improvements over state-of-the-art methods in both automatic and human evaluations.


Title
Martin writes a book

Outline
Martin wrote a book about his life experience.

Story
Martin was an older man with a lot of life experience. He wanted to put it all in a book. So he decided to sit down and write one. He wrote a book about all the things that happened in his life. He went on to sell his book and made a lot of money from it.

Figure 1: Neural story generation aims to generate a story given a title. The training data only contains pairs of titles and stories. Our work treats the outline that bridges the title and the story as a latent variable. Note that our work differs from previous work in that the outlines are represented as natural language sentences explainable to humans rather than real-valued vectors.

1 Introduction

Story generation, which aims to generate coherent stories automatically, is an important task in natural language processing (NLP). While early efforts have focused on symbolic and logical planning (McIntyre and Lapata, 2009; Riedl and Young, 2010; Martens et al., 2014) and case-based reasoning (Turner, 1994; Gervás et al., 2005), neural story generation (NSG) that leverages neural networks to generate stories has received increasing attention recently (Roemmele, 2016; Yao et al., 2019; Martin et al., 2018; Fan et al., 2018; Xu et al., 2018; Subramanian et al., 2018; Moryossef et al., 2019; Goldfarb-Tarrant et al., 2019).

Despite its rapid development, NSG still suffers from an "*information gap*" problem: it is hard to generate a long output given a short input. Figure 1 shows an example. Given a title "Martin writes a book", NSG needs to generate a coherent and fluent story accordingly. As the story is much longer than the title, it is challenging for NSG to ensure that the story is centered on the title and remains thematically consistent. This problem has been considered as one of the key challenges in story generation (Wiseman et al., 2017).

Although a number of authors have endeavored to address this problem by introducing high-level plots (i.e., outlines) to bridge titles and stories (Fan et al., 2018; Xu et al., 2018; Drissi et al., 2018; Yao et al., 2019) (see Section 2.1 for details), there still remain two unsolved problems. First, it is hard to learn the latent outlines. As the training data only contains titles and stories, outlines are unobserved. While it is natural to use a latent variable model (Kim et al., 2018) and treat outline as a latent variable, representing outlines as real-valued vectors makes it difficult for humans to interpret how outlines connect titles and stories. On the contrary, representing outlines as natural language sentences improves interpretability but faces the exponential search space of outlines.

Second, the lack of exposure to degenerate outlines impairs the generation performance. Drissi et al. (2018) indicate that their approach is incapable of dealing with poorly generated outlines because the generation model is only trained on good outlines in a separate way. When the generated outline is erroneous, the generation model is unlikely to be able to recover from the mistake.

In this work, we propose a method for learning to predict explainable high-level plots for NSG to alleviate the two aforementioned problems. In our model, the outline unobserved on the training data is treated as a latent variable. Given a training set that contains only pairs of titles and stories, we adopt an external summarization model to instruct the latent variable model to learn how to generate outlines. We also use a joint training objective to avoid explicit enumeration of exponentially many outlines and make the model exposed to predicted high-level plots (see Eq. (4)). Taking advantage of the generated outline to fill the information gap between the title and story, our approach can produce better stories compared with existing methods. Experiments on two widely used datasets show that our approach achieves significant improvements over existing methods in both automatic and human evaluations.

2 Related Work

Our work is closely related to two lines of research: hierarchical neural story generation and variational neural networks.

2.1 Hierarchical Neural Story Generation

Neural story generation has received increasing attention in recent years (Roemmele, 2016; Jain et al., 2017; Harrison et al., 2017; Wang et al., 2018b; Martin et al., 2018; Xu et al., 2018; Fan et al., 2018). Unlike other sequence-to-sequence learning tasks such as machine translation (Bahdanau et al., 2015; Vaswani et al., 2017) and dialogue generation (Serban et al., 2016), in which the input and output sequences are often of similar lengths, one major difficulty in neural story generation is that the output sequence is much longer than the input sequence. As a result, hierarchical models for neural story generation have been intensively studied recently (Xu et al., 2018; Fan et al., 2018; Drissi et al., 2018; Subramanian et al., 2018; Moryossef et al., 2019; Goldfarb-Tarrant et al., 2019; Yao et al., 2019; Fan et al., 2019).

Our work is closest to the hierarchical model proposed by Drissi et al. (2018), which also uses an outline as a high-level plot to reduce the information gap between input and output sequences. The major difference is that we treat outline generation as a learning task while Drissi et al. (2018) use an outline produced by an off-the-shelf text summarizer. Our experiments reveal that learning to generate outlines significantly improves over directly using the output of an off-the-shelf text summarizer.

This work also resembles the method proposed by Xu et al. (2018) because both skeletons and outlines are learned from data automatically. The difference is that we use outlines to bridge titles and stories while Xu et al. (2018) focus on using skeletons to better capture the dependencies between sentences in stories.

Our approach is also related to the method proposed by Yao et al. (2019), which generates a storyline from title to help generate the story. The main difference is that the outlines in our method consists of natural sentences while the model proposed by Yao et al. (2019) learns to generate a sequence of keywords as the storyline.

2.2 Variational Neural Networks

Variational neural networks (Graves, 2011; Kingma, 2013; Mnih and Gregor, 2014) have been widely used for many NLP tasks in recent years. Bowman et al. (2016) propose a generative model that uses a latent variable to learn the global feature for sentence representation. Zhang et al. (2016) design a variational encoder-decoder framework that leverages a latent variable to model the underlying semantics of source sentences. Calixto et al. (2018) incorporate image features to neural machine translation using latent variables.

Our work differs from prior studies on variational neural networks in that the latent variable is represented as discrete symbols rather than continuous vectors. This makes latent variables in our model interpretable to humans. It is easy for us to observe how the model learns the latent variables during training (see Table 6). This is beneficial for analyzing and understanding neural network based NSG models.

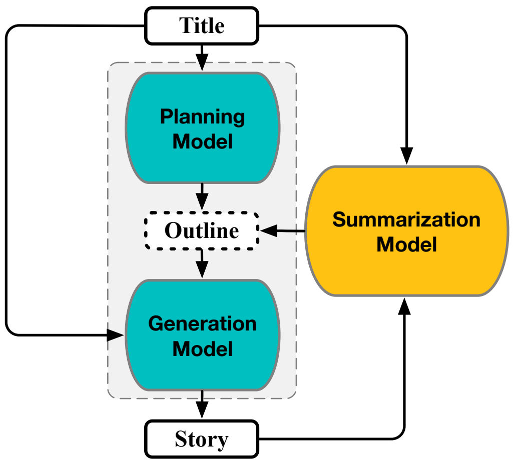


Figure 2: An overview of our latent variable model for NSG. Note that the summarization model is only used in the training phase while the planning model and the generation model are involved in both the training and inference phases.

3 Approach

Figure 2 shows an overview of our model. Given a title, we use a *planning model* to generate an outline, from which a *generation model* runs to produce the story (Section 3.1). As outlines are unobserved on the training data, we use a *summarization model* to “teach” the planning model how to generate outlines. We use the outlines produced by off-the-shelf text summarizers to pre-train the three models, which are then jointly trained on the training data (Section 3.2). Finally, the learned planning and generation models can be used to generate new stories for unseen titles (Section 3.3).

3.1 Modeling

The Latent Variable Model

Let $\mathbf{x} = x_1 \dots x_I$ be a title that contains I words, which is the input of neural story generation. We use $\mathbf{y} = y_1 \dots y_J$ to denote a story that contains J words, which is the output of neural story generation. As the stories in most widely used datasets such as ROCStories (Mostafazadeh et al., 2016) and VIST (Huang et al., 2016) are usually very short as shown in Figure 1, we assume there is only a one-sentence outline $\mathbf{z} = z_1 \dots z_K$ that contains K words for the entire story.

The latent variable model for neural story generation is given by

$$P(\mathbf{y}|\mathbf{x}; \boldsymbol{\theta}, \gamma) = \sum_{\mathbf{z}} \underbrace{P(\mathbf{z}|\mathbf{x}; \boldsymbol{\theta})}_{\text{planning}} \underbrace{P(\mathbf{y}|\mathbf{x}, \mathbf{z}; \gamma)}_{\text{generation}}, \quad (1)$$

where $P(\mathbf{z}|\mathbf{x}; \boldsymbol{\theta})$ is the *planning model* parameterized by $\boldsymbol{\theta}$ and $P(\mathbf{y}|\mathbf{x}, \mathbf{z}; \gamma)$ is the *generation model* parameterized by γ.

The Planning Model

The planning model is used to generate an outline given a title:

$$P(\mathbf{z}|\mathbf{x}; \boldsymbol{\theta}) = \prod_{k=1}^{K} P(z_k|\mathbf{x}, \mathbf{z}_{<k}; \boldsymbol{\theta}), \quad (2)$$

where $\mathbf{z}_{<k} = z_1 \dots z_{k-1}$ is a partial outline.

We use Transformer (Vaswani et al., 2017), which is the state-of-the-art sequence-to-sequence model for neural machine translation, to build the planning model. The title is the input of Transformer and the outline is the output.

The Generation Model

The generation model is used to generate a story given a title and an outline:

$$P(\mathbf{y}|\mathbf{x}, \mathbf{z}; \gamma) = \prod_{j=1}^{J} P(y_j|\mathbf{x}, \mathbf{z}, \mathbf{y}_{<j}; \gamma), \quad (3)$$

where $\mathbf{y}_{<j} = y_1 \dots y_{j-1}$ is a partial story.

The architecture of the generation model is similar to that of the planning model except that the input of the generation model has two sources, namely the title $\mathbf{x}$ and the outline $\mathbf{z}$. We expect that the generation model is able to take advantage of the information contained in the outline to guide the story generation. Inspired by (Zhang et al., 2018), which proposes a strategy to utilize the document-level context to improve sentence-level neural machine translation, we extend the vanilla sequence-to-sequence framework to a multi-source architecture that can make use of both titles and outlines. Figure 3 shows the architecture of the generation model. As indicated by the red arrows, the outline is not only used to learn a better representation of the title, but also participates in the generation of the story. Zhang et al. (2018) show that such an architecture is capable of effectively exploiting all sources of input. Note that the model reduces to the standard title-to-story architecture if the yellow components related to outlines are removed.

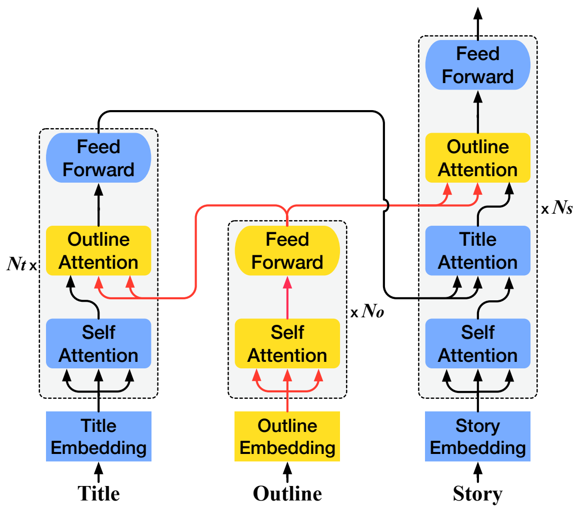


Figure 3: The architecture of the generation model.

3.2 Training

Training Objective

It is challenging to train the latent variable model in Eq. (1) because there are exponentially many possible outlines for a given title. As the full enumeration of all outlines is impossible and approximations have to be made, it is necessary to leverage useful external information to facilitate the learning of latent variables. The training objective of our model is given by

$$\begin{aligned} & J(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}, \gamma, \boldsymbol{\phi}) \\ = \; & \mathbb{E}_{\mathbf{z}|\mathbf{x};\boldsymbol{\theta}}\Big[\log P(\mathbf{y}|\mathbf{x}, \mathbf{z}; \gamma)\Big] \\ & -\mathrm{KL}\Big(\underbrace{P(\mathbf{z}|\mathbf{x}, \mathbf{y}; \boldsymbol{\phi})}_{\text{summarization}}\,\big|\big|P(\mathbf{z}|\mathbf{x}; \boldsymbol{\theta})\Big), \quad (4) \end{aligned}$$

where $P(\mathbf{z}|\mathbf{x}, \mathbf{y}; \boldsymbol{\phi})$ is the *summarization model* that generates the outline $\mathbf{z}$ based on the title $\mathbf{x}$ and the story $\mathbf{y}$.

The first term of the right-hand side of Eq. (4) is the expectation of the generation probability, which aims to make the model generate better stories using the titles and the outlines produced by the planning model. The second term is the Kullback-Leibler divergence between the summarization model and the planning model, which suggests that the summarization model is used to “teach” the planning model how to generate outlines. This term is also seen as a posterior regularization (Ganchev et al., 2010) where the summarization model act as the posterior that gives supervision for the planning model.

Note that Eq. (4) is similar but different from the lower bound widely used in variational inference (Kingma, 2013; Rezende et al., 2014). The expectation in Eq. (4) would be defined with respect to $P(\mathbf{z}|\mathbf{x}, \mathbf{y}; \boldsymbol{\phi})$ instead of $P(\mathbf{z}|\mathbf{x}; \boldsymbol{\theta})$ in variational inference. We did not directly use variational inference because it is the planning model rather than the summarization model that is used during inference (see Section 3.3). Therefore, we are interested in the expectation with respect to the planning model. Our experiment shows that our approach improves over variational inference on the NSG task (see Table 4).

The summarization model can be defined as

$$P(\mathbf{z}|\mathbf{x}, \mathbf{y}; \boldsymbol{\phi}) = \prod_{k=1}^{K} P(z_k|\mathbf{x}, \mathbf{y}, \mathbf{z}_{<k}; \boldsymbol{\phi}). \quad (5)$$

We use an architecture similar to that in Figure 3. The major difference is that the title and the story serve as the input and the outline is the output.

Pre-training

Pre-training has proven to be important for providing a good starting point for parameter estimation in natural language processing (Devlin et al., 2019). We follow Drissi et al. (2018) to use an off-the-shelf text summarizer to generate raw outlines, which are used to pre-train our models.

Let $\{\langle \mathbf{x}^{(s)}, \mathbf{y}^{(s)}\rangle\}_{s=1}^{S}$ be the training data that contains S pairs of titles and stories. We can use an off-the-shelf text summarizer to generate a set of outlines $\{\tilde{\mathbf{z}}^{(s)}\}_{s=1}^{S}$ given all the stories. We use these outlines to pre-train the planning, generation, and summarization models by maximum likelihood estimation:

$$\boldsymbol{\theta}_0 = \operatorname*{argmax}_{\boldsymbol{\theta}} \left\{ \sum_{s=1}^{S} \log P(\tilde{\mathbf{z}}^{(s)}|\mathbf{x}^{(s)}; \boldsymbol{\theta}) \right\}, \quad (6)$$

$$\gamma_0 = \operatorname*{argmax}_{\gamma} \left\{ \sum_{s=1}^{S} \log P(\mathbf{y}^{(s)}|\mathbf{x}^{(s)}, \tilde{\mathbf{z}}^{(s)}; \gamma) \right\}, \quad (7)$$

$$\boldsymbol{\phi}_0 = \operatorname*{argmax}_{\boldsymbol{\phi}} \left\{ \sum_{s=1}^{S} \log P(\tilde{\mathbf{z}}^{(s)}|\mathbf{x}^{(s)}, \mathbf{y}^{(s)}; \boldsymbol{\phi}) \right\}. \quad (8)$$

Parameter Estimation

Both of the two terms in the right-hand side of Eq. (4) require summing up all outlines to calculate expectations. Note that KL divergence can also be seen as an expectation. Due to the exponential space of outlines, we use Monte Carlo sampling to approximate the expectations. We draw M samples $\{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(M)}\}$ from the planning model $P(\mathbf{z}|\mathbf{x}; \boldsymbol{\theta})$ and draw N samples $\{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(N)}\}$ from the summarization model $P(\mathbf{z}|\mathbf{x}, \mathbf{y}; \boldsymbol{\phi})$.

As a result, the training objective can be approximated as

$$
\begin{aligned}
& J(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}, \gamma, \boldsymbol{\phi}) \\
\approx\ & \frac{1}{M} \sum_{m=1}^{M} \log P(\mathbf{y}|\mathbf{x}, \mathbf{z}^{(m)}; \gamma) \\
& -\frac{1}{N} \sum_{n=1}^{N} \log \frac{P(\mathbf{z}^{(n)}|\mathbf{x}, \mathbf{y}; \boldsymbol{\phi})}{P(\mathbf{z}^{(n)}|\mathbf{x}; \boldsymbol{\theta})}. \qquad (9)
\end{aligned}
$$

Given a training set $\{\langle \mathbf{x}^{(s)}, \mathbf{y}^{(s)} \rangle\}_{s=1}^{S}$, the final loss function for training our model can be formulated as

$$L(\boldsymbol{\theta}, \gamma, \boldsymbol{\phi}) = -\sum_{s=1}^{S} J(\mathbf{x}^{(s)}, \mathbf{y}^{(s)}, \boldsymbol{\theta}, \gamma, \boldsymbol{\phi}). \quad (10)$$

Initialized with the pre-trained model parameters $\boldsymbol{\theta}_0$, γ_0, and $\boldsymbol{\phi}_0$, our latent variable model can be optimized as follows:

$$\hat{\boldsymbol{\theta}}, \hat{\gamma}, \hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\theta}, \gamma, \boldsymbol{\phi}}{\operatorname{argmin}} \Big\{ L(\boldsymbol{\theta}, \gamma, \boldsymbol{\phi}) \Big\}. \quad (11)$$

3.3 Inference

The generation process is divided into two steps. Given a new title $\mathbf{x}$, our model first generates an outline using the planning model:

$$\hat{\mathbf{z}} = \underset{\mathbf{z}}{\operatorname{argmax}} \Big\{ P(\mathbf{z}|\mathbf{x}; \hat{\boldsymbol{\theta}}) \Big\}. \quad (12)$$

Then, the generation model is used to generate the story given the title and the outline:

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}} \Big\{ P(\mathbf{y}|\mathbf{x}, \hat{\mathbf{z}}; \hat{\gamma}) \Big\}. \quad (13)$$

Note that the summarization model is only involved in training and is not used in inference.

4 Experiments

4.1 Datasets

We evaluated our approach on two widely used datasets: the ROCStories dataset (Mostafazadeh et al., 2016) and the VIST dataset (Huang et al., 2016).

The ROCStories dataset contains 98,161 short stories as the training set and additional 1,871 stories as validation and test sets, respectively. Since only the training set of the ROCStories dataset contains titles, which we need as input, we split the original training set into 8:1:1 for training, validation, and testing. Therefore, the training set we used in the experiments contains 78,529 pairs of titles and stories, the validation set and the test set both contain 9,816 pairs.

The VIST dataset contains pairs of photo sequences and corresponding coherent hand-crafted narratives of events through time. Sentences with the same story ID in the original dataset are concatenated to form a story. Since all the stories in this dataset have no titles, we followed Xu et al. (2018) to use the first sentence of an original story as the title and the remaining sentences as the story. The processed dataset contains 40,155, 4,990, and 5,055 pairs of titles and stories for training, validation, and testing, respectively.

4.2 Baselines

We compared our approach with the following NSG methods:

1. DIRECT (Roemmele, 2016): Directly generating a story given a title using sequence-to-sequence models.

2. SKELETON (Xu et al., 2018): First generating a skeleton and then expanding the skeleton into a complete sentence until the whole story has been generated.

3. HIERARCHICAL (Fan et al., 2018): First generating a prompt and then generating a story

based on the prompt.

4. SEPARATE (Drissi et al., 2018): First generating an outline and then generating a story based on the outline using two separate modules.

5. PLANNED (Yao et al., 2019): First generating a storyline and then generating a story based on the title and the storyline.

For DIRECT, we used Transformer (Vaswani et al., 2017) to directly map titles into stories. For SKELETON, we directly used the code [1] released by Xu et al. (2018). For HIERARCHICAL, we directly used the code [2] released by Fan et al. (2018). For SEPARATE, we used two separate Transformers to model the transition from titles to outlines and the transition from outlines to stories, respectively. For PLANNED, we directly used the code [3] released by Yao et al. (2019), and more specifically, we adopted the static planning scheme. Please refer to (Yao et al., 2019) for more details.

For convenience, we refer to our approach as LATENT. The planning, generation, and summarization models are built on top of the Transformer architecture. The encoders of the planning and generation models are composed of a stack of two identical layers while the encoder of the summarization model comprises a stack of six identical layers. All the decoders of the three models consist of a stack of six identical layers. We set M and N in Eq. (9) to 1, which is good enough in our experiments. Our approach also used the text summarizer proposed by Nenkova and Vanderwende (2005) to extract outlines from stories and pre-train our models.

We used beam search to produce stories and set the beam size to 4 in the inference phase. For fair comparisons, we set the hidden size and embedding size to 256, and maximum vocabulary size to 50K for all the baselines and our approach. We set the dimension of the feed-forward network to 1,024 for DIRECT, SEPARATE, and our method.

4.3 Evaluation Metrics

Both automatic and human evaluations are conducted in our experiments.

Automatic Evaluation Metrics

Following existing methods (Guan et al., 2019; Wang et al., 2018a), we use BLEU-4 (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), ROUGE-L (Lin, 2004), and model perplexity as automatic evaluation metrics. BLEU and ROUGE-L encourage exact match between ground-truth text and generated text by n-gram overlap, and METEOR also uses WordNet stems and synonyms. Perplexity is designed to evaluate the quality of language models because it reflects how fluently the model can produce the correct next word given the preceding words. Smaller perplexity indicates better performance.

Human Evaluation Metrics

As Wang et al. (2018b) state that there are no perfect automatic evaluation metrics to quantify the quality of generated stories, it is important to use human evaluation metrics. In our experiments, we used three human evaluation metrics: *fluency*, *relevance*, and *coherence*. Fluency measures whether each sentence in the story is grammatically correct. Relevance measures whether the story is related to the title. Coherence evaluates whether the story is coherent. Each metric score ranges from 1 to 10. Higher score indicates better performance.

For each test set, we randomly selected 100 titles and obtained 600 stories from all the six methods (i.e., five baselines and our method). Then, we asked annotators to score the stories in terms of fluency, relevance, and coherence. Note that all the annotators have linguistic background. The stories

[1]https://github.com/lancopku/Skeleton-Based-Generation-Model

[2]https://github.com/pytorch/fairseq

[3]https://bitbucket.org/VioletPeng/language-model

from six methods were shuffled. The correspondences between methods and stories were invisible to the annotators. Each story was scored by three annotators. The scores were averaged to give the final score. We also computed the geometric mean of the above three metrics to measure the overall performance.

4.4 Evaluation Results

Automatic Evaluation

Table 1 lists the results of automatic evaluation on the ROCStories and VIST datasets. SKELETON generally performs worse than other baselines except that it obtains lower perplexities than SEPARATE on both datasets. Surprisingly, although DIRECT does not use any high-level plot, it outperforms other baselines in terms of BLEU, METEOR, and perplexity. One possible reason is that the stories in both datasets are relatively short. The superiority of hierarchical models over direct mapping might increase when generating very long stories. Our approach significantly outperforms all baselines in terms of all listed metrics on both datasets ($p < 0.01$) [4].

Human Evaluation

Table 2 shows the results of human evaluation on the ROCStories and VIST datasets. While all the six methods achieve close fluency scores because of the use of neural networks, the gaps on relevance and coherence are much larger. On the ROCStories dataset, our approach achieves significantly better results than all baselines in terms of fluency, relevance, and G-Mean except that SEPARATE obtains a slightly higher coherence score than our approach. On the VIST dataset, our method achieves the highest scores on all the metrics by large margins. These differences are statistically significant ($p < 0.01$).

[4]We did the statistical significance tests by paired bootstrap resampling (Koehn, 2004).

Ablation Study

Table 3 shows the results of ablation study. Our approach reduces to DIRECT by removing the planning model. Note that the summarization model is also disabled in this case. Only discarding the summarization model leads to the worst results. This suggests that the summarization model is critical for improving the performance of our approach: it is difficult to learn a good planning model without the guidance of the summarization model. Besides, the performance of our method also drops without pre-training, which indicates the effectiveness of pre-training in our approach.

Comparison with Variational Inference

Table 4 shows the comparison between our approach and variational inference on the ROCStories dataset. We find that training by variation inference deteriorates the performance of our model. One possible reason is that in variational inference the generation model is not exposed to the outlines generated by the planning model during training. During inference, however, the generation model takes the outline output by the planning model as one input. This mismatch between training and inference seems to be harmful for the NSG task. In contrast, the training objective in Eq. (4) results in significantly better performance in terms of all metrics ($p < 0.01$).

Effect of Title Length

Figure 4 shows the effect of title length on the ROCStories dataset. In the test set, there are 2,546 stories for 1-word titles, 4,110 stories for 2-word titles, 1,989 stories for 3-word titles, 808 stories for 4-word titles, and 252 stories for 5-word titles. The average BLEU score was calculated for each length. We find that all methods achieve higher average BLEU scores for longer titles. This is because the information gap between title and story can be reduced if the title contains more information. Our approach outperforms all baselines over

Model	ROCStories				VIST			
	B-4 ↑	MET ↑	R-L ↑	PPL ↓	B-4 ↑	MET ↑	R-L ↑	PPL ↓
DIRECT	3.01	9.95	22.09	14.83	2.88	8.27	20.59	38.72
SKELETON	1.45	7.43	19.82	29.17	0.96	5.82	18.20	46.53
HIERARCHICAL	2.09	9.65	20.66	25.79	2.55	9.34	21.10	38.05
SEPARATE	2.55	9.44	21.47	31.30	2.75	8.04	20.40	58.54
PLANNED	2.87	9.98	22.30	28.67	2.61	9.63	20.79	40.43
LATENT	**3.70**	**10.61**	**23.06**	**12.33**	**3.35**	**10.03**	**21.76**	**36.34**

Table 1: Automatic evaluation on the ROCStories and VIST datasets. We use "B-4", "MET", "R-L", and "PPL" to denote BLEU-4, METEOR, ROUGE-L, and perplexity scores, respectively.

Model	ROCStories				VIST			
	F ↑	R ↑	C ↑	M ↑	F ↑	R ↑	C ↑	M ↑
DIRECT	6.47	5.06	5.98	5.86	5.20	4.67	5.32	5.07
SKELETON	6.00	3.93	4.06	4.76	5.22	4.47	5.01	4.91
HIERARCHICAL	5.69	4.26	4.44	4.84	5.42	4.85	5.39	5.22
SEPARATE	6.47	4.35	**6.15**	5.73	5.93	4.56	5.75	5.44
PLANNED	6.35	4.43	5.37	5.44	5.86	4.94	5.52	5.45
LATENT	**6.78**	**5.80**	6.13	**6.25**	**6.43**	**5.69**	**6.35**	**6.17**

Table 2: Human evaluation on the ROCStories and VIST datasets. We use "F" to denote fluency, "R" to denote relevance, "C" to denote coherence, and "M" to denote the geometric mean of fluency, relevance, and coherence.

Model	BLEU	METEOR	ROUGE-L
LATENT	**3.70**	**10.61**	**23.06**
– PM	3.01	9.95	22.09
– SM	2.90	9.83	21.97
– PT	3.39	10.32	22.49

Table 3: The results of ablation study. We use "PM" to denote the planning model, "SM" to denote the summarization model, and "PT" to denote pre-training.

Training	BLEU	METEOR	ROUGE-L
VI	3.41	10.23	21.47
LATENT	**3.70**	**10.61**	**23.06**

Table 4: Comparison with variational inference on the ROCStories dataset. We use "VI" to denote variational inference (see Section 3.2).

all lengths and the gaps seem to be enlarged on longer titles.

Effect of Training Corpus Size

Figure 5 shows the effect of training corpus size. To investigate how the performance is influenced by the size of training data, we randomly extracted part of the data from the original training set of the ROCStories dataset in the proportion of 10%, 30%, 50%, and 80%, respectively. It is clear that using more training data is beneficial for improving BLEU scores. We find that our approach keeps the superiority over all baselines consistently over different sizes of training data.

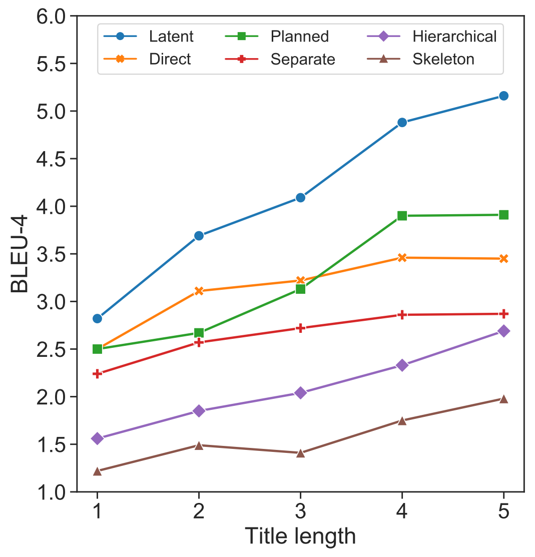


Figure 4: Effect of title length.

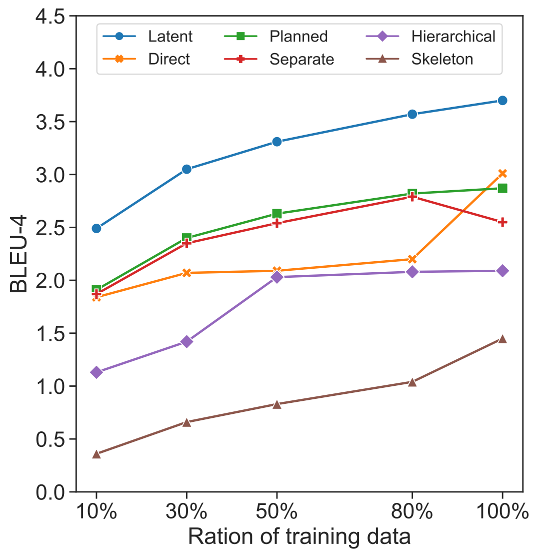


Figure 5: Effect of training corpus size.

Comparison of Generated Stories

Table 5 shows the stories generated by all methods given the title "Fainting in the store". The story generated by DIRECT includes a phrase "passed out" that is similar to "fainting" but ignores "store". The story generated by SKELETON is fluent and coherent but is totally unrelated to the title because both "fainting" and "store" do not appear in the story. Despite containing the keyword "store" that is related to the title, the story generated by PLANNED lacks the relevance to "fainting" and suffers from sentence repetition.

The story generated by HIERARCHICAL shares the same problem with that of SKELETON because it is neither related to "fainting" nor related to "store". Using outlines to provide a high-level plot, SEPARATE generated a fluent and coherent story. However, there is no "fainting" in the story. One possible reason is that SEPARATE uses an off-the-shelf text summarizer to extract an outline only from the story. As the stories in the two datasets are relatively short, direct mapping between titles and stories achieves much better performance than other baselines. In addition, DIRECT avoids the error propagation problem (i.e., noisy plots inevitably leads to degenerate stories) by direct mapping.

By learning to generate outlines guided by the summarizion model, our approach is able to generate a fluent story centered on the title. A problem with the story is that "ice cream" seems to be irrelevant and should be replaced with "milk". How to model such cross-sentence dependencies in stories is another key problem in story generation. In this work, such dependencies are modeled using self-attention (Vaswani et al., 2017). It is necessary to develop more powerful mechanisms to further improve the coherence at the story level.

Learned Outlines during Training

Table 6 shows the outlines learned by the summarization and planning models at different iterations during training. We find the outlines learned by the summarization model often remain identical

Model	Story
DIRECT	Tom was at the bar. He was drinking a bit too much. He got a bit too drunk. He got a bit too drunk and **passed out**. He had to be taken back home.
SKELETON	Patty was at a college. She was looking for a new camera. Patty saw a woman. She was looking for a ring.
HIERARCHICAL	I was on a road trip with my family. We had been looking for hours. The train was on the side of the road. The driver had to come to the step to get it. The driver did not want to move back in that car.
SEPARATE	Maria was at the grocery **store**. She was shopping for a new brand. Suddenly, she realized that she forgot her wallet. She looked everywhere for it. Luckily, she was able to find it in her pocket.
PLANNED	I went to the **store** to buy a new bag. I wanted to buy a new one. I went to the **store** and bought a new one. I went to the **store** to buy a new one. I bought it and went to the **store**.
LATENT	Patty was at the grocery **store**. She was carrying a glass of milk. Patty **fell to the ground**. The ice cream fell on the ground. Patty had to go to the hospital to get stitches.

Table 5: Stories generated by all methods given a title “Fainting in the store”. Words that are closely related to the title are highlighted in bold.

Title	Back to School	
Story	Cabot’s summer break was finally over. She had a wonderful summer this year. She can’t wait to see her friends at school. It is hard to get to sleep the night before classes start. After a good breakfast, she was finally an official eighth grader.	
Outlines	Iteration 0	(S) Cabot’s summer break was finally over. (P) The kids went to school.
	Iteration 32K	(S) Cabot’s summer break was finally over. (P) She was excited to go back to school after the summer holidays.

Table 6: The outlines learned by the summarization and planning models during training. We use “(S)” and “(P)” to denote the outlines generated by the summarization and planning models, respectively.

during training. On the contrary, the outlines generated by the planning model keep changing during training. While the predicted outline is “The kids went to school.” at iteration 0 in the beginning, the prediction is changed to “She was excited to go back to school after the summer holidays”. According to Eq. (4), the planning model is involved in both terms of the training objective, which suggests that the outline generated by the planning model should not only be close to that of the summarization model, but also enable the generation model to generate the ground-truth story. We find that the outline generated by the planning model at iteration 32K contains more useful information about the story than that of the summarization model. Besides filling the information gap between titles and stories, these predicted outlines are interpretable to humans and help them better understand how NSG model works.

5 Conclusion and Future Work

We have proposed a method for learning to predict explainable plots for neural story generation for generating fluent, coherent, and reasonable stories. Different from existing methods, our model is capable of learning to automatically produce an explainable high-level plot to bridge the information gap between the title and story. Experiments on two benchmark datasets show that our method outperforms state-of-the-art approaches to neural story generation in both automatic and human evaluations.

In this work, we have focused on using a one-sentence outline to bridge the information gap between a title and a short story. How to model the cross-sentence dependencies in a long story still remains a severe challenge. It is important to develop more effective mechanisms to improve story-level coherence. We plan to extend our method to multiple outline generation for generating very long stories in the future.

References

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR 2015*.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL 2005 Workshop*.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. arXiv:1511.06349.

Iacer Calixto, Miguel Rios, and Wilker Aziz. 2018. Latent variable model for multi-modal translation. arXiv:1811.00357.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT 2019*.

Mehdi Drissi, Olivia Watkins, and Jugal Kalita. 2018. Hierarchical text generation using an outline. arXiv:1810.08802.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hiearchical neural story generation. In *Proceedings of ACL 2018*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2019. Strategies for structuring story generation. In *Proceedings of ACL 2019*.

Kuzman Ganchev, Jennifer Gillenwater, Ben Taskar, et al. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11(Jul):2001–2049.

Paolo Gervás, Belén Diaz-Agudo, Federico Peinado, and Raquel Hervás. 2005. Story plot generation based on CBR. *Journal of Knowledge-based Systems*, 18(4-5):235–242.

Seraphina Goldfarb-Tarrant, Haining Feng, and Nanyun Peng. 2019. Plan, write, and revise: an interactive system for open-domain story generation. In *Proceedings of NAACL-HLT 2019*.

Alex Graves. 2011. Practical variational inference for neural networks. In *Proceedings of NeurIPS 2011*.

Jian Guan, Yansen Wang, and Minlie Huang. 2019. Story ending generation with incremental encoding and commonsense knowledge. In *Proceedings of AAAI 2019*.

Brent Harrison, Christopher Purdy, and Mark Riedl. 2017. Toward automated story generation with markov chain monte carlo methods and deep neural networks. In *Proceedings of AAAI 2017*.

Ting-Hao Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Aishwarya Agrawal, Jacob Devlin, Ross Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, C. Lawrence Zitnick, Devi Parikh, Lucy Vanderwende, Michel Galley, and Margaret Mitchell. 2016. Visual storytelling. In *Proceedings of NAACL-HLT 2016*.

Parag Jain, Priyanka Agrawal, Abhijit Mishra, Mohak Sukhwani, Anirban Laha, and Karthik Sankaranarayanan. 2017. Story generation from sequence of independent short descriptions. In *Proceedings of KDD 2017 Workshop*.

Yoon Kim, Sam Wiseman, and Alexander Rush. 2018. A tutorial on deep latent variable models of natural language. arXiv:1812.06834.

Max Kingma, Diederik P.and Welling. 2013. Auto-encoding variational bayes. arXiv:1312.6114.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP 2004*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Proceedings of the ACL 2004 Workshop*.

Chris Martens, João F. Ferreira, Anne-Gwenn Bosser, and Marc Cavazza. 2014. Generative story worlds as linear logic programs. In *Proceedings of AAAI 2014*.

Lara J. Martin, Prithviraj Ammanabrolu, Xinyu Wang, William Hancock, Shruti Singh, Brent Harrison, and Mark O. Riedl. 2018. Event representations for automated story generation with deep neural nets. In *Proceedings of AAAI 2018*.

Neil McIntyre and Mirella Lapata. 2009. Learning to tell tales: A data-driven approach to story generation. In *Proceedings of ACL 2009*.

Andriy Mnih and Karol Gregor. 2014. Neural variational inference and learning in belief networks. In *Proceedings of ICML 2014*.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019. Step-by-step: Separating planning from realization in neural data-to-text generation. In *Proceedings of NAACL-HLT 2019*.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushment Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of NAACL-HLT 2016*.

Ani Nenkova and Lucy Vanderwende. 2005. The impact of frequency on summarization. Technical report, Microsoft Research.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL 2002*.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of ICML 2014*.

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

Melissa Roemmele. 2016. Writing stories with help from recurrent neural networks. In *Proceedings of AAAI 2016*.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of AAAI 2016*.

Sandeep Subramanian, Sai Rajeswar Mudumba, Alessandro Sordoni, Adam Trischler, Aaron C Courville, and Chris Pal. 2018. Towards text generation with adversarially learned neural outlines. In *Advances in NeuIPS 2018*.

Scott R. Turner. 1994. *The Creative Process: A Computer Model of Storytelling*. Lawrence Erlbaum Associates.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NeurIPS 2017*.

Qingyun Wang, Zhihao Zhou, Lifu Huang, Spencer Whitehead, Boliang Zhang, Heng Ji, and Kevin Knight. 2018a. Paper abstract writing through editing mechanism. In *Proceedings of ACL 2018*.

Xin Wang, Wenhu Chen, Yuan-Fang Wang, and William Yang Wang. 2018b. No metrics are perfect: Adversarial reward learning for visual storytelling. In *Proceedings of ACL 2018*.

Sam Wiseman, Stuart M. Shieber, and Alexander M. Rush. 2017. Challenges in data-to-document generation. In *Proceedings of EMNLP 2017*.

Jingjing Xu, Xuancheng Ren, Yi Zhang, Qi Zeng, Xiaoyan Cai, and Xu Sun. 2018. A skeleton-based model for promoting coherence among sentences in narrative story generation. In *Proceedings of EMNLP 2018*.

Lili Yao, Nanyun Peng, Ralph Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-and-write: Towards better automatic storytelling. In *Proceedings of AAAI 2019*.

Biao Zhang, Deyi Xiong, jinsong su, Hong Duan, and Min Zhang. 2016. Variational neural machine translation. In *Proceedings of EMNLP 2016*.

Jiacheng Zhang, Huanbo Luan, Maosong Sun, Feifei Zhai, Jingfang Xu, Min Zhang, and Yang Liu. 2018. Improving the transformer translation model with document-level context. In *Proceedings of EMNLP 2018*.