Cue Me In: Content-Inducing Approaches to Interactive Story Generation

Faeze Brahman[†], **Alexandru Petrusca**[†], **and Snigdha Chaturvedi**[‡]
[†]Department of Computer Science and Engineering, University of California, Santa Cruz
[‡]Department of Computer Science, University of North Carolina at Chapel Hill
{fbrahman, apetrusc}@ucsc.edu snigdha@cs.unc.edu

Abstract

Automatically generating stories is a challenging problem that requires producing causally related and logical sequences of events about a topic. Previous approaches in this domain have focused largely on one-shot generation, where a language model outputs a complete story based on limited initial input from a user. Here, we instead focus on the task of interactive story generation, where the user provides the model mid-level sentence abstractions in the form of cue phrases *during* the generation process. This provides an interface for human users to guide the story generation. We present two content-inducing approaches to effectively incorporate this additional information. Experimental results from both automatic and human evaluations show that these methods produce more topically coherent and personalized stories compared to baseline methods.

1 Introduction

Automatic story generation requires composing a coherent and fluent passage of text about a sequence of events. Prior studies on story generation mostly focused on symbolic planning (Lebowitz, 1987; Pérez y Pérez and Sharples, 2001; Porteous and Cavazza, 2009; Riedl and Young, 2010) or case-based reasoning (Gervás et al., 2005) that heavily relied on manual knowledge engineering.

Recent state-of-the-art methods for story generation (Martin et al., 2018; Clark et al., 2018a) are based on sequence-to-sequence models (Sutskever et al., 2014) that generate a story in one go. In this setting, the user has little control over the generated story.

On the other hand, when humans write, they incrementally edit and refine the text they produce. Motivated by this, rather than generating the entire story at once, we explore the problem of interactive story generation. In this setup, a user can provide the model mid-level sentence abstractions in the form of *cue phrases* as the story is being generated. Cue phrases enable the user to inform the system of what they want to happen next in the story and have more control over what is being generated. To achieve our goal, this paper primarily focuses on approaches for smoothly and effectively incorporating user-provided cues. The schematic in Fig. 1 illustrates this scenario: the system generates the story one sentence at a time, and the user guides the content of the next sentence using cue phrases. We note that the generated sentences need to fit the context, and also be semantically related to the provided cue phrase.

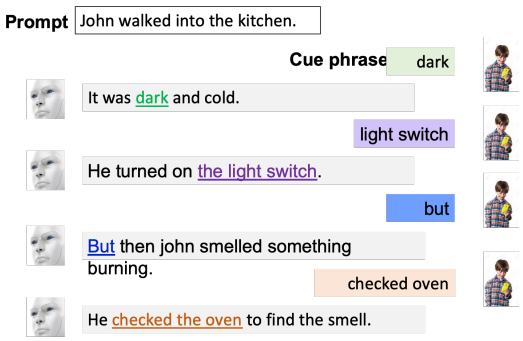


Figure 1: Interactive story generation: the user inputs the first sentence of the story (prompt), and provides guiding cue phrases as the system generates the story one sentence at a time.

A fundamental advantage of using this framework as opposed to a fully automated one is that it can provide an interactive interface for human users to incrementally supervise the generation by giving signals to the model throughout the story generation process. This human-computer collaboration can result in generating richer and personalized stories. In particular, this field of research can be used in addressing the literacy needs of learners with disabilities and enabling children to explore

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing, pages 588–597

creative writing at an early age by crafting their own stories.

In this paper, we present two content-inducing approaches based on the Transformer Network (Vaswani et al., 2017) for interactively incorporating external knowledge when automatically generating stories. Here, our external knowledge is in the form of cue phrases provided by the user to enable interaction, but can readily be replaced with knowledge accessible through other means[1]. Specifically, our models fuse information from the story context and cue phrases through a hierarchical attention mechanism. The first approach, *Cued Writer*, employs two independent encoders (for incorporating context and cue phrases) and an additional attention component to capture the semantic agreement between the cue phrase and output sentence. The second approach, *Relevance Cued Writer*, additionally measures the relatedness between the context and cue phrase through a context-cue multi-head unit. In both cases, we introduce different attention units in a single end-to-end neural network.

Our automatic and human evaluations demonstrate that the presented models outperform strong baselines and can successfully incorporate cues in generated stories. This capability is one step closer to an interactive setup, and unlike one-shot generation, it lets users have more control over the generation. Our contributions are twofold:

- Two novel content-inducing approaches to incorporate additional information, in this case cue phrases, into the generation phase.
- Experiments demonstrating utility of content-inducing approaches using automatic and human evaluations.

2 Related Work

Automatic story generation is a longstanding problem in AI, with early work dating back to the 1970s based on symbolic planning (Lebowitz, 1987; Pérez y Pérez and Sharples, 2001; Porteous and Cavazza, 2009; Riedl and Young, 2010) and case-based reasoning using ontologies (Gervás et al., 2005). Li et al. (2013) extended prior works toward learning domain models (via corpus and/or crowdsourcing) to support open story generation about any topic.

[1]For example, the user-provided cues can be replaced by the outputs of an automatic planner. Our models are flexible enough to work in other setups.

With the advent of deep learning there has been a major shift towards using seq2seq models (Sutskever et al., 2014; Bahdanau et al., 2015) for various text generation tasks, including storytelling (Roemmele, 2016; Jain et al., 2017; Hu et al., 2020). However, these models often fail to ensure coherence in the generated story. To address this problem, Clark et al. (2018a) incorporated entities given their vector representations, which get updated as the story unfolds. Similarly, Liu et al. (2020) proposed a character-centric story generation by learning character embeddings directly from the corpus. Fan et al. (2018) followed a two-step process to first generate the premise and then condition on that to generate the story. Yu et al. (2020) proposed a multi-pass CVAE to improve wording diversity and content consistency.

Previous work has explored the potential of creative writing with a machine in the loop. Clark et al. (2018b) found that people generally enjoy collaborating with a machine. Traditional methods proposed to write stories collaboratively using a case-based reasoning architecture (Swanson and Gordon, 2012). Recent work (Roemmele and Gordon, 2015) extended this to find relevant suggestions for the next sentence in a story from a large corpus. Other methods proposed GUI and tools to facilitate co-creative narrative generation (Manjavacas et al., 2017; Kapadia et al., 2015). Unlike us, these approaches explore the value of and tools for interaction rather than designing methods for incorporating user input into the model.

Another line of research decomposes story generation into two steps: story plot planning and plot-to-surface generation. Previous work produces story-plans based on sequences of events (Martin et al., 2018; Tambwekar et al., 2019; Ammanabrolu et al., 2020), critical phrases (Xu et al., 2018) or both events and entities (Fan et al., 2019). Yao et al. (2019) model the story-plan as a sequence of keywords. They proposed *Static* and *Dynamic* paradigms that generate a story based on these story-plans. Goldfarb-Tarrant et al. (2019) adopted the *static* model proposed in Yao et al. (2019) to supervise story-writing.

A major focus of these works is on generating a coherent plan for generating the story. In contrast, our contribution is complementary since we do not focus on *planning* but on *generation*. We present approaches to effectively incorporate external knowledge in the form of cue-phrases during

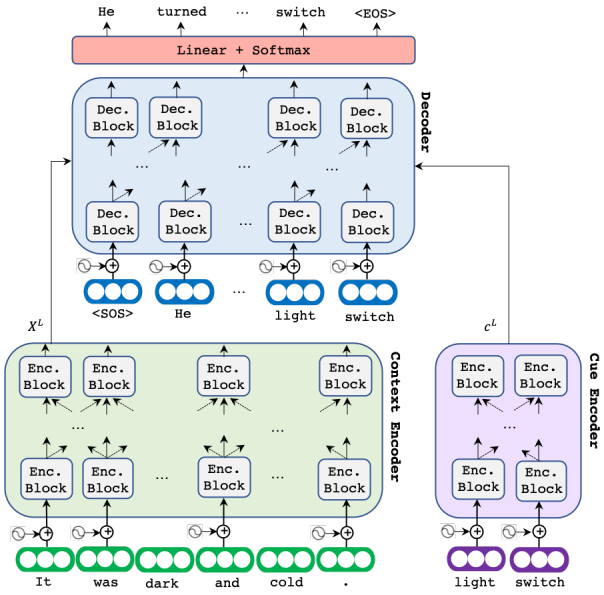


Figure 2: Overall model architecture for *Cued Writer* and *Relevance Cued Writer*.

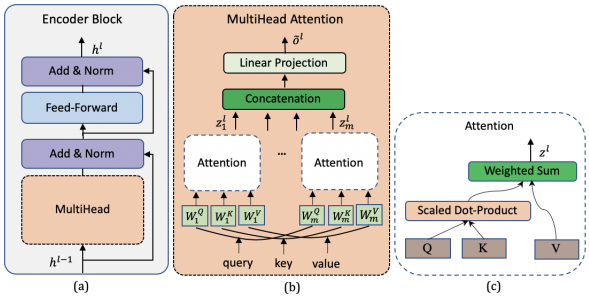


Figure 3: **(a)** Encoder Block consists of MultiHead and FFN. **(b)** MultiHead Attention. **(c)** Attention Module.

generation, and conduct extensive experiments to compare our models with those of Yao et al. (2019) by modifying them to work in our setup.

3 Interactive Story Generation

We design models to generate a story one sentence at a time. Given the generated context so far (as a sequence of tokens) $X = \{x_1, ..., x_T\}$, and the cue phrase for the next sentence $c = \{c_1, ..., c_K\}$, our models generate the tokens of the next sentence of the story $Y = \{y_1, ..., y_M\}$. We train the models by minimizing the cross-entropy loss:

$$L_\theta = -\sum_{i=1}^{M} \log P(y_i|X, c, \theta) \quad (1)$$

Here, θ refers to model parameters. Note that when generating the n-th sentence, the model takes the first $n-1$ sentences in the story as the context along with the cue phrase.

In the rest of this section, we describe our two novel content-inducing approaches for addressing the interactive story generation task: the *Cued Writer*, and the *Relevance Cued Writer*. These models share an overall encoder-decoder based architecture shown in Fig. 2. They adopt a dual encoding approach where two separate but architecturally similar encoders are used for encoding the context (Context Encoder represented in the green box) and the cue phrase (Cue Encoder represented in the purple box). Both these encoders advise the Decoder (represented in the blue box), which in turn generates the next sentence. The two proposed models use the same encoding mechanism (described in § 3.1) and differ only in their decoders (described in § 3.2).

3.1 Encoder

Our models use the Transformer encoder introduced in Vaswani et al. (2017). Here, we provide a generic description of the encoder architecture followed by the inputs to this architecture for the Context and Cue Encoders in our models.

Each encoder layer l contains architecturally identical Encoder Blocks, referred to as EncBlock (with unique trainable parameters). Fig. 3(a) shows an Encoder Block which consists of a Multi-Head attention and an FFN that applies the following operations:

$$\tilde{o}^l = \text{MultiHead}(h^{l-1}) \quad (2a)$$
$$o^l = \text{LayerNorm}(\tilde{o}^l + h^{l-1}) \quad (2b)$$
$$\tilde{h}^l = \text{FFN}(o^l) \quad (2c)$$
$$h^l = \text{LayerNorm}(\tilde{h}^l + o^l) \quad (2d)$$

Where MultiHead represents Multi-Head Attention (described below), FFN is a feed-forward neural network with ReLU activation (LeCun et al., 2015), and LayerNorm is a layer normalization (Ba et al., 2016). In the rest of the paper, LayerNorm (also shown as `Add & Norm` in figures) is always applied after MultiHead and FFN, but we do not explicitly mention that in text or equations for simplicity.

Multi-Head Attention The multi-head attention, shown in Fig. 3(b), is similar to that used in Vaswani et al. (2017). It is made of multiple Attention heads, shown in Fig. 3(c). The Attention head has three types of inputs: the query sequence,

$Q \in R^{n_q \times d_k}$, the key sequence, $K \in R^{n_k \times d_k}$, and the value sequence, $V \in R^{n_v \times d_k}$. The attention module takes each token in the query sequence and attends to tokens in the key sequence using a scaled dot product. The score for each token in the key sequence is then multiplied by the corresponding value vector to form a weighted sum:

$$\textsc{Attn}(\mathrm{Q}, \mathrm{K}, \mathrm{V}) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (3)$$

For each head, all Q, K, and V are passed through a head-specific projection prior to the attention being computed. The output of a single head is:

$$H_i = \textsc{Attn}(QW_i^Q, KW_i^K, VW_i^V) \quad (4)$$

Where Ws are head-specific projections. Attention heads H_i are then concatenated:

$$\textsc{MultiH}(\mathrm{Q}, \mathrm{K}, \mathrm{V}) = [H_i; ...; H_m]W^O \quad (5)$$

Where W^O is an output projection. In the encoder, all query, key, and value come from the previous layer and thus:

$$\textsc{MultiHead}(h^{l-1}) = \textsc{MultiH}(h^{l-1}, h^{l-1}, h^{l-1}) \quad (6)$$

Encoder Input The Encoder Blocks described above form the constituent units of the Context and Cue Encoders, which process the context and cue phrase respectively. Each token in the context, x_i, and cue phrase, c_i, is assigned two kinds of embeddings: *token embeddings* indicating the meaning and *position embeddings* indicating the position of each token within the sequence. These two are summed to obtain individual input vectors, X^0, and c^0, which are then fed to the first layer of Context and Cue encoders, respectively. Thereafter, new representations are constructed through layers of encoder blocks:

$$X^{l+1} = \textsc{EncBlock}(X^l, X^l, X^l) \quad (7a)$$

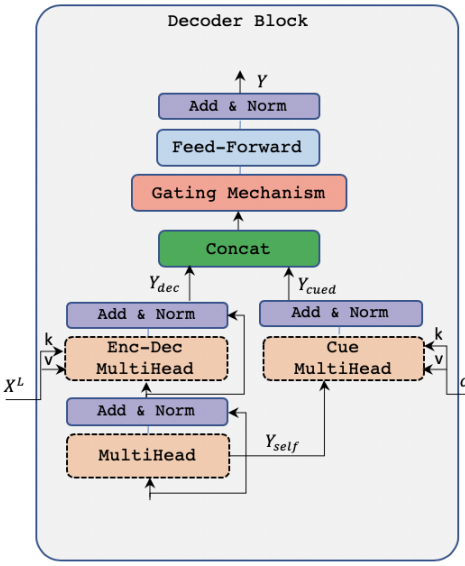
$$c^{l+1} = \textsc{EncBlock}(c^l, c^l, c^l) \quad (7b)$$

where $l \in [0, L-1]$ denotes different layers. In Eqn. 7a and 7b, the output of the previous layer's Encoder Block is used as Q, K, and V input for the multi-head attention of the next block.

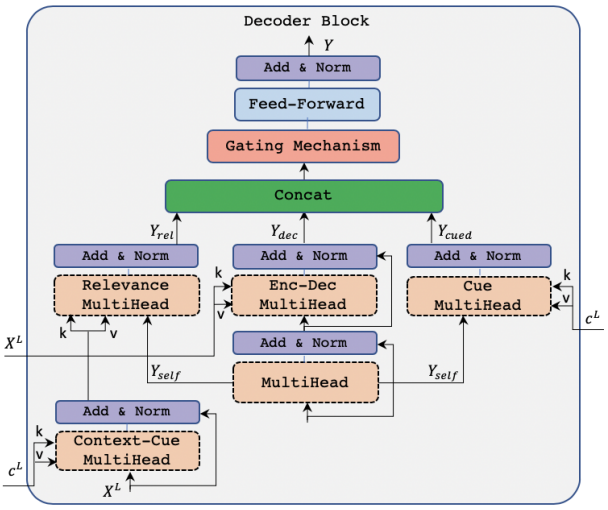
3.2 Content-Inducing Decoders

We now describe the decoders for our models.

Cued Writer The main intuition behind our first model, *Cued Writer*, is that since cue phrases indicate users' expectations of what they want to see in the next sentence of the story, they should be used by the model *at the time of generation*, i.e., in the decoder. Below, we describe the decoder used by the *Cued Writer*.



(a) Cued Writer



(b) Rel. Cued Writer

Figure 4: Decoder architectures. X^L and c^L are the outputs of the top-layers of the Context and Cue encoders respectively, and K and V are the corresponding keys and values.

After processing the two types of inputs in the Context and Cue Encoders, the model includes their final encoded representations (X^L and c^L) in the decoder. The decoder consists of L layers with architecturally identical Decoder Blocks. Each Decoder Block contains `Enc-Dec MultiHead` and the `Cue MultiHead` units (see Fig. 4(a)), which let the decoder to focus on the relevant parts of the context and the cue phrase, respectively.

Given Y^0 as the word-level embedding represen-

tation for the output sentence, our Decoder Block is formulated as:

$$Y^{l+1}_{self} = \text{MULTIH}(Y^l, Y^l, Y^l) \quad (8a)$$

$$Y^{l+1}_{dec} = \text{MULTIH}(Y^{l+1}_{self}, X^L, X^L) \quad (8b)$$

$$Y^{l+1}_{cued} = \text{MULTIH}(Y^{l+1}_{self}, c^L, c^L) \quad (8c)$$

Eqn. 8a is standard self-attention, which measures the intra-sentence agreement for the output sentence and corresponds to the `MultiHead` unit in Fig. 4(a). Eqn. 8b, describing the `Enc-Dec MultiHead` unit, measures the agreement between context and output sentence, where queries come from the decoder Multi-Head unit (Y_{self}), and the keys and values come from the top layer of the context encoder (X^L). Similarly, Eqn. 8c captures the agreement between output sentence and cue phrase through `Cue MultiHead` unit. Here, keys and values come from the top layer of the Cue encoder (c^L).

Lastly, we adapt a gating mechanism (Sriram et al., 2018) to integrate the semantic representations from both Y_{dec} and Y_{cued} and pass the resulting output to FFN function:

$$g^{l+1} = \sigma(W_1[Y^{l+1}_{dec}; Y^{l+1}_{cued}]) \quad (9a)$$

$$Y^{l+1}_{int} = W_2(g^{l+1} \circ [Y^{l+1}_{dec}; Y^{l+1}_{cued}]) \quad (9b)$$

$$Y^{l+1} = \text{FFN}(Y^{l+1}_{int}) \quad (9c)$$

the representation from Y_{dec} and Y_{cued} are concatenated to learn gates, g. The gated hidden layers are combined by concatenation and followed by a linear projection with the weight matrix W_2.

Relevance Cued Writer The decoder of *Cued Writer* described above captures the relatedness of the context and the cue phrase to the generated sentence but does not study the relatedness or relevance of the cue phrase to the context. We incorporate this relevance in the decoder of our next model, *Relevance Cued Writer*. Its Decoder Block (shown in Fig. 4(b)) is similar to that of *Cued Writer* except for two additional units: the Context-Cue and Relevance MultiHead units. The intuition behind the `Context-Cue MultiHead` unit (Eqn. 10a) is to characterize the relevance between the context and the cue phrase, so as to highlight the effect of words in the cue phrase that are more relevant to the context thereby promoting topicality and fluency. This relevance is then provided to the decoder using the `Relevance MultiHead` unit (Eqn. 10b):

$$X^{l+1}_{rel} = \text{MULTIH}(X^L, c^L, c^L) \quad (10a)$$

$$Y^{l+1}_{rel} = \text{MULTIH}(Y^{l+1}_{self}, X^{l+1}_{rel}, X^{l+1}_{rel}) \quad (10b)$$

We fuse the information from all three sources using a gating mechanism and pass the result to FFN:

$$g^{l+1} = \sigma(W_1[Y^{l+1}_{dec}; Y^{l+1}_{cued}; Y^{l+1}_{rel}]) \quad (11a)$$

$$Y^{l+1}_{int} = W_2(g^{l+1} \circ [Y^{l+1}_{dec}; Y^{l+1}_{cued}; Y^{l+1}_{rel}]) \quad (11b)$$

$$Y^{l+1} = \text{FFN}(Y^{l+1}_{int}) \quad (11c)$$

Finally, for both models, a linear transformation and a softmax function (shown in Fig. 2) is applied to convert the output produced by the stack of decoders to predicted next-token probabilities:

$$P(y_i|y_{<i}, X, c, \theta) = \text{softmax}(Y^L_i W_y) \quad (12)$$

where $P(y_i|y_{<i}, X, c, \theta)$ is the likelihood of generating y_i given the preceding text ($y_{<i}$), context and cue, and W_y is the token embedding matrix.

4 Empirical Evaluation

4.1 Dataset

We used the ROCStories corpus (Mostafazadeh et al., 2016) for experiments. It contains $98,161$ five-sentence long stories with a rich set of causal/temporal sequences of events. We held out 10% of stories for validation and 10% for test set.

4.2 Baselines

SEQ2SEQ Our first baseline is based on a LSTM sentence-to-sentence generator with attention (Bahdanau et al., 2015). In order to incorporate user-provided cue phrases, we concatenate context and cue phrase with a delimiter token (<$>) before passing it to the encoder.

DYNAMIC This is the Dynamic model proposed by Yao et al. (2019) modified to work in our setting. For a fair comparison, instead of generating a plan, we provide the model with cue phrases and generate the story one sentence at a time.

STATIC The STATIC model (Yao et al., 2019) gets all cue phrases at once to generate the entire story[2]. By design, it has additional access to all, including future, cue phrases. Our models and other baselines do not have this information.

VANILLA To verify the effectiveness of our content-inducing approaches, we use a Vanilla Transformer as another baseline and concatenate context and cue phrase using a delimiter token.

[2]We used the implementation available at: https://bitbucket.org/VioletPeng/language-model/

Models	PPL (↓)	BLEU-1 (↑)	BLEU-2 (↑)	BLEU-3 (↑)	GM (↑)	Repetition-4 (↓)
DYNAMIC (Yao et al., 2019)	29.49	30.05	9.16	4.59	0.73	44.36
STATIC (Yao et al., 2019)	20.81	33.25	9.64	4.77	0.75	26.26
SEQ2SEQ	20.97	33.91	10.01	3.09	0.82	33.23
VANILLA	15.78	40.30	16.09	7.19	0.89	20.87
Cued Writer	14.80	41.50	16.72	7.25	0.92	**15.08**
Rel. Cued Writer	**14.66**	**42.65**	**17.33**	**7.59**	**0.94**	16.23

Table 1: Automatic evaluation results. Our models outperform all baselines across all metrics ($p < 0.05$).

4.3 Training details

Following previous work (Vaswani et al., 2017), we initialize context encoders and decoders with 6 layers (512 dimensional states and 8 attention heads). Our models contain 3-layer encoders for encoding cue phrases (all other specifications are the same). For the position-wise feed-forward networks, we use 2048 dimensional inner states. We use the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.0001 and residual, embedding, and attention dropouts with a rate of 0.1 for regularization. Models are implemented in PyTorch, trained for 30 epochs with early stopping on validation loss.

Cue-phrases for Training and Automatic Evaluation: For training all models, we need cue phrases, which are, in principle, to be entered by a user. However, to scale model training, we automatically extracted cue phrases from the target sentences in the training set using the previously proposed RAKE algorithm (Rose et al., 2010). It is important to note that cue phrases can represent a variety of information, and many other methods can be used to extract them for training purposes. For example, topic words, distinctive entities or noun phrases in the sentence, the headword in the dependency parse of the sentence, etc.

Our automatic evaluations were done on a large-scale, and so we followed a similar approach for extracting cue-phrases.

Cue-phrases for Human Evaluation: In the interest of evaluating the interactive nature of our models, cue-phrases were provided manually during our interactive evaluations[3].

General Statistics on Cue-phrases: Automatically extracted cue phrases has the vocabulary size of 22,097, and 6,189 on the train and test set, respectively with the average 10% coverage over the entire target sentence. Cue-phrases are typically 1-2 words. Comparing user-provided vs automatically extracted cue-phrases, the average length of user-provided cue-phrases in interactive evaluation is 1.56, with a vocabulary size of 206, whereas these numbers are 1.59 and 214 for their corresponding automatically extracted cue phrases.

[3]We left the definition of cue-phrase open-ended to enable flexibility in user interaction. They are typically 1-2 words.

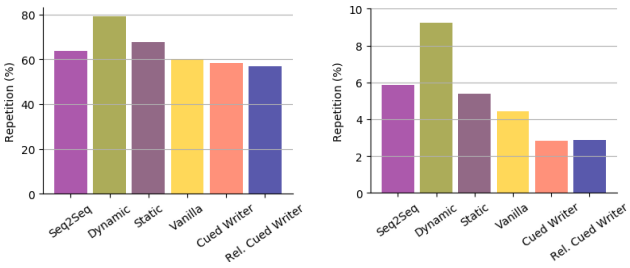


Figure 5: Inter-story (left) and Intra-story (right) repetition scores. The proposed models have better scores.

4.4 Automatic Evaluation

Following previous credible works (Martin et al., 2018; Fan et al., 2018), we compare various methods using Perplexity and BLEU (Papineni et al., 2002) on the test set. We reported BLEU-n for n=1, 2, 3. From Table 1, we can see that both our models outperform DYNAMIC and STATIC by large margins on perplexity and BLEU scores. The proposed models are also superior to the SEQ2SEQ and VANILLA baseline on both measures. Comparing the last two rows of Table 1, we also see an additive gain from modeling the relevance in *Rel. Cued Writer*. All improvements are statistically significant (approximate randomization (Noreen, 1989), $p < 0.05$).

To evaluate how well the story generation model incorporates the cues, we use an embedding-based greedy matching score (GM) (Liu et al., 2016). The score measures the relatedness of the generated story with cues by greedily matching them with each token in a story based on the cosine similarity of their word embeddings (Yao et al., 2019). We can see from the 5th column in Table 1 that our models generate stories that are more related to the cue phrases.

Prompt (first sentence): Jordan was watching TV on her couch.
Cue phrases: watch football - change channel - comedy show - very funny
She was trying to watch football on TV. Then she went to change channel. Finally, she decided to watch a comedy show. She saw the comedy that was playing and didn't like.
Cue phrases: soccer - cook - order pizza - tasty dinner
Her brother was playing in a soccer. She wasn't able to cook. Instead, she ordered pizza. Her brother was happy with the tasty dinner.

Table 2: Example of stories generated in interactive evaluation using two models given the same prompt and different set of cue-phrase.

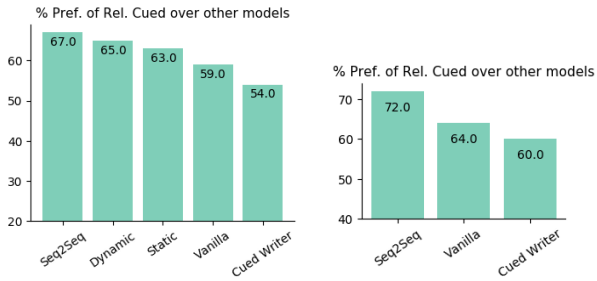
Previous works have shown that neural generation models suffer from repetition issue; and so we additionally evaluate the models using repetition-4 which measures the percentage of generated stories that repeat at least one 4-gram (Shao et al., 2019) and inter- and intra-story repetition scores (Yao et al., 2019). A lower value is better for these scores. The result of repetition-4 is reported in the last column of Table 1. The proposed models significantly outperform all baselines, and among the two *Cued Writer* is better. Inter and intra repetition scores are depicted in Fig. 5. Our two proposed models are almost comparable on these metrics but they show a general superior performance compared to all baselines. In particular, *Rel. Cued Writer* achieves a significant performance increase of 16% and 46% on these scores over the stronger model of Yao et al. (2019)[4].

4.5 Human Evaluation

Automatic metrics cannot evaluate all aspects of open-ended text generation (Fan et al., 2018), and so we also conduct several human evaluations.

Interactive Evaluation In this experiment, human subjects compare our best model, *Rel. Cued Writer*, with the strongest baseline from the automatic evaluations (VANILLA) in an interactive, real-time setup.

For robust evaluation, it is essential that the users generate a wide variety of stories. Since generating different prompts (first sentence) requires creativity on the part of human judges and can be challenging, we provided participants with initial prompts that were randomly selected from the test set. For each prompt, the participants generated stories using both models by interactively providing cue-phrases[5]. They were then asked to choose which story they prefer. Participants preferred stories generated by *Rel. Cued Writer* over VANILLA in 57.5% of the cases (80 stories in total, $p \sim 0.1$).



(a) (b)

Figure 6: Human evaluations on story-level (left) and sentence-level (right). We find that human judges preferred stories generated by Rel. Cued Writer.

Judges also rated the stories in terms of fluency and coherence on a 5-point Likert scale. *Rel. Cued Writer* achieved a higher fluency score of 4.22 compared with 3.80 achieved by VANILLA. VANILLA attained a slightly higher coherence score (3.40 vs. 3.35). On manually inspecting the generated stories, we found that our model generates longer sentences (avg. 9.18 words) with more complex language, whereas VANILLA generated relatively shorter sentences (avg. 7.46 words) which might improve coherence.

This experiment is promising but inconclusive because for the same prompt, the participants could provide different sets of cue-phrases for different models, resulting in generated stories that are too different to be comparable (Table 2 shows an example). This led us to conduct the following more controlled evaluations.

Story-level Evaluation In this experiment, we again make pairwise comparisons, but both models are provided the same prompts, and sets of cue phrases[6]. 3 judges evaluated 100 pairs of stories (in shuffled order)[7].

Fig. 6(a) shows the percentage of preference for our stronger model, *Rel. Cued Writer*, over the baselines. Judges prefer our model over all other baselines. Also, judges preferred *Rel. Cued Writer* over *Cued Writer*, which demonstrates the effec-

[4]Note that the result of our SEQ2SEQ baseline is not directly comparable with that of Inc-S2S in (Yao et al., 2019), since we included cue phrases as additional input whereas Inc-S2S generate the whole story conditioned on the title.

[5]We included instructions and examples for participants. The order of the presentation of the models was random. The judges were self-identified native English speakers.

[6]Cue-phrases were extracted automatically to alleviate any biases, then curated manually.

[7]For all AMT evaluations, we only allow judges who have at least 500 previous jobs, and at least 97% approval rate.

Prompt	I was coughing very badly. | **Cue phrases** notice blood - emergency room - waited - okay
DYNAMIC	I noticed that I had a headache. I went to the doctor. The doctor told me that I had to go to **the doctor**. The doctor said I was fine. **[waited]**
STATIC	**She** noticed a huge blood pressure. She went to the doctor. **The doctor told her to stop.** She waited for a while. **[okay]**
SEQ2SEQ	I noticed blood and **noticed blood** in the closet. **Now I have emergency room** for **emergency room**. I found it and I waited until the next day. I feel guilty and now i don't have to do it again. **[okay]**
VANILLA	I noticed blood in the bathroom. I had to go to the emergency room. I waited in the bathroom for a while. Luckily **the doctor was okay.**
Cued Writer	I noticed blood in my mouth. I had to go to the emergency room. I waited there a few hours. I felt okay.
Rel Cued Writer	I noticed blood on my shirt. I went to the emergency room. I waited for the doctor. He came and I was okay.
Prompt	Harry really wanted to see the new Star Wars movie. | **Cue phrases** opening day - ticket - entire movie - really
DYNAMIC	He went to the theater to show up. He bought a ticket. He had a great time. **He had a great time. [opening day] [entire movie]**
STATIC	One day, **she** was in a hurry to get to the theater. she wasn't sure if she would like to go. She was excited to try the new movie. **She was so excited** to see the movie. **[opening day] [ticket]**
SEQ2SEQ	The day **day** was opening day and his family was **opening** the **opening day**. Harry had bought a ticket and the **ticket** wasn't very good. The entire movie was very happy. Harry became **very really** disappointed.
VANILLA	On opening day, Harry was very nervous. He bought a ticket **to** the theater. He bought **Harry** ticket **tickets** to the theater. He really didn't like the movie. **[entire movie]**
Cued Writer	On opening day, he went to the theater . He bought a ticket at the theater. The entire movie was great. He really was excited.
Rel Cued Writer	He decided to watch it on opening day. He got to the theater and got a ticket. He watched the entire movie. He was really excited about it.

Table 3: Sample stories generated by different models. We highlight in different color the **[missing]** cue phrase, **incoherent or unfluent**, and **repetitive** parts of each story. We see that compared to baselines, our models correctly mention cue phrases and generate better stories.

tiveness of the additional Context-Cue and Relevance Multi-Head units. All improvements are statistically significant (app. rand., $p < 0.05$).

Sentence-level Evaluation We also performed a more fine-grained evaluation of the models by evaluating generated sentences while the model is generating a story. The generated sentences are evaluated in light of the (incomplete) story. Specifically, we provide an (incomplete) story passage and a manually provided cue phrase to the two models to generate the next sentence. We asked human judges to identify which of the two sentences is better based on their fluency and semantic relevance to (1) the input (incomplete) story and (2) the cue phrase. We did this experiment for a set of 100 randomly selected stories (400 sentences. 3 different judges evaluated each sentence pair. Fig. 6(b) shows that the *Rel. Cued Writer* model was preferred over SEQ2SEQ and VANILLA in 72% and 64% of the cases, respectively. Comparing the two proposed models, we again see additive gain by modeling Cue-Context relevance. All improvements are statistically significant (app. rand., $p < 0.001$).

5 Qualitative Results and Error Analysis

Table 3 presents examples of stories generated by different models for the same prompt and cue phrases. We highlight the [missing] cue phrases, incoherent or unfluent, and repetitive parts of each story. Note that we did not highlight [missing], if the model mentions part of the cue phrase or incorporates it semantically. As we observe, all of the baselines suffered from several issues; however, our novel content inducing approaches generate more causally related sentences, which fit the given prompt and cue phrases more naturally.

off-topic: Kelly and her friends went to a new ice-cream shop. They decided to try the new flavors. They all tried on many different restaurants. To their surprise, they thought it tasted good. They were glad to find one online.
Not-logically-consistent: Avery received a homework assignment due in two weeks. He immediately read it. When he turned it in, he made schedule. He completed tasks and turned it in time. When he finished early, he was disappointed.
non-coreferent-pronouns: Rob has never been on a rollercoaster. They go on all the way to six flags. He got on with a free ticket. Rob joined the rollercoaster. There was a long line of people in the line.

Table 4: Examples of errors made by our model.

We also manually reviewed 50 stories, generated from our models and analyzed common errors. Table 4 shows sample stories that depict different types of errors including "getting off-topic", "not-logically-connected" and "non-coreferent pronouns". The last type of error represents the cases where the model generates pronouns that do not refer to any previously mentioned entity. The examples demonstrate that there are still many challenges in this domain.

6 Conclusion and Future Work

This paper explored the problem of interactive storytelling, which leverages human and computer collaboration for creative language generation. We presented two content-inducing approaches that take user-provided inputs as the story progresses and effectively incorporate them in the generated text. Experimental results show that our methods outperform competitive baselines. However, there are several other significant aspects to be considered in story generation, such as modeling of discourse relations, and representation of key narrative elements, which lie beyond the scope of this investigation. Also, while we received encouraging feedback from users on this setup during the interactive evaluation, we did not explore important questions about user interfaces, design, and human computer interaction. Future work can explore these questions and also explore other forms of natural language interaction.

References

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

Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*.

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

Elizabeth Clark, Yangfeng Ji, and Noah A. Smith. 2018a. Neural text generation in stories using entity representations as context. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2250–2260.

Elizabeth Clark, Anne Spencer Ross, Chenhao Tan, Yangfeng Ji, and Noah A. Smith. 2018b. Creative writing with a machine in the loop: Case studies on slogans and stories. In *Proceedings of the 23rd International Conference on Intelligent User Interfaces*, pages 329–340.

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.

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

Pablo Gervás, Belén Díaz-Agudo, Federico Peinado, and Raquel Hervás. 2005. Story plot generation based on CBR. *In Applications and Innovations in Intelligent Systems XII*, 28(1):33–46.

Seraphina Goldfarb-Tarrant, Haining Feng, and Nanyun Peng. 2019. Plan, write, and revise: an interactive system for open-domain story generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Demonstrations*, pages 89–97.

Junjie Hu, Yu Cheng, Zhe Gan, Jingjing Liu, Jianfeng Gao, and Graham Neubig. 2020. What makes a good story? Designing composite rewards for visual storytelling. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 7969–7976.

Parag Jain, Priyanka Agrawal, Abhijit Mishra, Mohak Sukhwani, Anirban Laha, and Karthik Sankaranarayanan. 2017. Story generation from sequence of independent short descriptions. *Workshop on Machine Learning for Creativity*.

Mubbasir Kapadia, Jessica Falk, Fabio Zünd, Marcel Marti, Robert W. Sumner, and Markus H. Gross. 2015. Computer-assisted authoring of interactive narratives. In *Proceedings of the 19th Symposium on Interactive 3D Graphics and Games*, pages 85–92.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*.

Micheal Lebowitz. 1987. Planning stories. *In Proceedings of the cognitive science society*, pages 234–242.

Yann LeCun, Y Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521:436–44.

Boyang Li, Stephen Lee-Urban, George Johnston, and Mark Riedl. 2013. Story generation with crowdsourced plot graphs. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, page 598–604.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132.

Danyang Liu, Juntao Li, Meng-Hsuan Yu, Ziming Huang, Gongshen Liu, Dongyan Zhao, and Rui Yan.

2020. A character-centric neural model for automated story generation. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 1725–1732.

Enrique Manjavacas, Folgert Karsdorp, Ben Burtenshaw, and Mike Kestemont. 2017. Synthetic literature: Writing science fiction in a co-creative process. In *Proceedings of the Workshop on Computational Creativity in Natural Language Generation,*, pages 29–37.

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 the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 868–875.

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.

Eric W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses: An Introduction*. Wiley New York.

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.

Rafael Pérez y Pérez and Mike Sharples. 2001. Mexica: A computer model of a cognitive account of creative writing. *Journal of Experimental Theoretical Artificial Intelligence*, 13(2):119–139.

Jullie Porteous and Mike Cavazza. 2009. Controlling narrative generation with planning trajectories: the role of constraints. *In Joint International Conference on Interactive Digital Storytelling*, pages 234–245.

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 the Thirtieth AAAI Conference on Artificial Intelligence*, page 4311–4312.

Melissa Roemmele and Andrew S. Gordon. 2015. Creative Help: A Story Writing Assistant. In *Interactive Storytelling*, pages 81–92.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text Mining: Applications and Theory*, pages 1 – 20.

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*, pages 3257–3268.

Anuroop Sriram, Heewoo Jun, Sanjeev Satheesh, and Adam Coates. 2018. Cold fusion: Training seq2seq models together with language models. In *6th International Conference on Learning Representations, ICLR 2018, Workshop Track Proceedings*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems*, pages 3104–3112.

Reid Swanson and Andrew S. Gordon. 2012. Say Anything: Using Textual Case-Based Reasoning to Enable Open-Domain Interactive Storytelling. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 2(3):16:1–16:35.

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

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 *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, pages 5998–6008.

Jingjing Xu, Yi Zhang, Qi Zeng, Xuancheng Ren, Xiaoyan Cai, and Xu Sun. 2018. A skeleton-based model for promoting coherence among sentences in narrative story generation. *In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4306–4315.

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

Meng-Hsuan Yu, Juntao Li, Danyang Liu, Bo Tang, Haisong Zhang, Dongyan Zhao, and Rui Yan. 2020. Draft and edit: Automatic storytelling through multi-pass hierarchical conditional variational autoencoder. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 1741–1748.