Generating Coherent Narratives by Learning Dynamic and Discrete Entity States with a Contrastive Framework

Jian Guan[1], Zhenyu Yang[2], Rongsheng Zhang[3], Zhipeng Hu[3] and Minlie Huang[1]*

[1]The CoAI group, DCST, Institute for Artificial Intelligence, State Key Lab of Intelligent Technology and Systems, Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing 100084, China.
[2]Guangdong OPPO Mobile Telecommunications Corp., Ltd.
[3]Fuxi AI Lab, NetEase Inc., Hangzhou, China.
j-guan19@mails.tsinghua.edu.cn; yangzhenyu@oppo.com; {zhangrongsheng, zphu}@corp.netease.com; aihuang@tsinghua.edu.cn

Abstract

Despite advances in generating fluent texts, existing pretraining models tend to attach incoherent event sequences to involved entities when generating narratives such as stories and news. We conjecture that such issues result from representing entities as static embeddings of superficial words, while neglecting to model their ever-changing states, i.e., the information they carry, as the text unfolds. Therefore, we extend the Transformer model to dynamically conduct entity state updates and sentence realization for narrative generation. We propose a contrastive framework to learn the state representations in a discrete space, and insert additional attention layers into the decoder to better exploit these states. Experiments on two narrative datasets show that our model can generate more coherent and diverse narratives than strong baselines with the guidance of meaningful entity states.

1 Introduction

Generating open-ended texts that maintain long-range coherence is important for myriad natural language generation applications such as narrative generation. The task requires, given a short input, creating a text with a sequence of coherent events involving several interleaved entities (e.g., characters, organizations, locations, etc.). As the text unfolds, the entities encounter different events, enrich their respective information, and accordingly change readers' expectations about them, thus playing a central role to make the text coherent (Grosz, Joshi, and Weinstein 1995). Arguably, the ability to dynamically predict unseen events attached to different entities is indispensable for generation (Henaff et al. 2017) but has not yet been widely investigated.

Typical generative models such as BART (Lewis et al. 2020) are trained to learn co-occurrence between tokens, which are represented as learnable embeddings. As exemplified in Table 1, BART can easily capture dependencies between common words such as "injured" and "funeral," but attaches incoherent event sequences to the involved entities. We conjecture that such issues arise from representing entities as no more than static embeddings of superficial words throughout whole texts, while neglecting to model the change of information these entities carry, i.e., their states (Ji et al. 2017). Since the same superficial words can co-occur with different events (and vice versa), it is necessary for generating coherent texts to model dependencies between events and specific states instead of word embeddings. For instance, in the first example of Table 1, after "Bobby's funeral", the model should learn to update its estimate of Bobby's state from being *alive* to *dead* and then attach such events to him as "lying in a coffin" instead of "going to a graveyard." Similarly, in the second example, Apple and Samsung transit to different states after "the court rule that Samsung violated the patent on all Apple's...," and the model should attach the event "fighting the ruling" to Samsung instead of Apple.

Example 1: ... After **Bobby is injured**, the scene flashes to **Bobby's funeral**. **Bobby then goes to a graveyard** with the family and begins to tell of a woman that ...
Example 2: ... The Tokyo District Court found **Samsung guilty** last week for "intentionally causing serious injury to the plaintiffs by copying, copying, or causing injury to a trademarked trademark." The court ruled that **Samsung violated the patent on all Apple's mobile devices**, including the iPhone and iPad ... **Apple has been fighting the ruling** since Wednesday ...

Table 1: Two examples generated by BART fine-tuned on the Wikiplots and CNN News datasets, respectively, where conflicting events are attached to some entities (in bold). Different entity mentions are marked in different colors.

This work proposes a generation model that incorporates dynamic entity states to improve coherence. We extend the Transformer decoder to update entity states after each sentence, which then serve to guide the subsequent decoding. We design a novel contrastive framework to learn the state representations. Instead of conditioning on continuous state representations for generation (Clark, Ji, and Smith 2018), we use a set of discrete state vectors to represent entity states, which are a natural fit for state transitions. Discrete states also encourage effective use of the latent space, and alleviate excessive focus on local and imperceptible details.

The contrastive framework is designed to pull the state of an entity close to events that can be attached to the en-

*Corresponding author

tity in the representation space. To abstract high-level event features, we adopt an external encoder to encode each sentence in a mini-batch to obtain the corresponding entity-aware event representations for different entities in the sentence. At the end of each sentence, we first learn to predict which entity to mention in the following sentence. We then learn the current state representation of the entity using a contrastive objective by regarding the representation of the following event attached to it as the positive and all others in the same mini-batch as negatives. During inference, we look up the closest state vector to the state representation in the pre-defined discrete latent space, and feed the state vector into the decoder along with the word embedding of the entity mention to guide the following sentence realization. The hidden states of the decoder get access to input entity states using not only the vanilla self-attention layer, but also a state attention layer inserted into each decoder block. The additional layer narrows the attention scope to only prefix entity states for explicitly modeling the dependencies between states and contextual events. In the training phase, we directly use the closest state vector to the following event representation as input to keep the parallel training efficiency. Our contributions are as follows:

I. We propose a novel generation model that learns dynamic and discrete **e**ntity state **r**epresentat**i**ons with a **c**ontrastive framework (ERIC). We equip the decoder with an additional attention layer to apply the entity states to guide decoding.

II. Experiments show that ERIC learns a set of meaningful entity state vectors corresponding to different clusters of events that can be attached to an entity, thus generating more coherent and informative texts than strong baselines[1].

2 Related Work

Narrative Generation Recent studies presented a series of multi-step generation models for narrative generation, which first planned intermediate sketches like keywords (Yao et al. 2019), semantic role labeling tags (Fan, Lewis, and Dauphin 2019) and keyword distributions (Kang and Hovy 2020), and then generated whole texts conditioned on them. Ji and Huang (2021) learned a sequence of discrete latent codes to abstract high-level discourse structures. Each code corresponds to a fixed-length span, which does not always agree with real text structures and makes it hard to model specific semantic dependencies. Some studies tried to incorporate external knowledge or reasoning models to guide commonsense story generation (Guan et al. 2020; Xu et al. 2020; Ammanabrolu et al. 2021), which may lack generalization to other domains such as news. Another line improved coherence by learning high-level representations of prefix sentences (Li, Luong, and Jurafsky 2015; Guan et al. 2021), which does not emphasize the central role of entities.

Prior studies on state tracking commonly adopted an external memory for state reads and writes. Ji et al. (2017) and Clark, Ji, and Smith (2018) updated the state of an entity conditioned on previous hidden outputs when encountering its mention, which does not apply to the parallel architecture of Transformer for training. Rashkin et al. (2020) and Papalampidi, Cao, and Kocisky (2022) performed state updates at the paragraph and chunk level, respectively, to alleviate but not eliminate the issue. In contrast, ERIC learns entity states through entity-aware event representations derived from an external encoder during training, thus well adapting to popular pretraining models, and achieving more frequent state updates. Furthermore, our work presents the first study for learning discrete entity states.

Hierarchical Transformers It is necessary to capture the hierarchical structure of natural language texts (Ribeiro et al. 2020). Previous work employed hierarchical attention networks for document classification (Yang et al. 2016) and machine translation (Miculicich et al. 2018). Guo et al. (2019) adopted a hierarchy network to build document embeddings on top of sentence embeddings for document mining. Similarly, HIBERT (Zhang, Wei, and Zhou 2019) incorporated a hierarchical architecture to BERT (Devlin et al. 2019) for document summarization. These models aim to enhance encoders for modeling long inputs and are less adaptive for generation. Recent studies tried to shorten sequences in intermediate decoder blocks to learn high-level representations for text classification (Dai et al. 2020) and generation (Nawrot et al. 2021). Hu et al. (2022) incorporated dynamically planned bag-of-words representations to guide the text realization without considering dependencies between words. In comparison, the proposed contrastive framework enables the learned entity states to serve as a sentence-level guidance for generation.

Contrastive Learning Contrastive learning has become popular in unsupervised visual and textual representation learning (Hadsell, Chopra, and LeCun 2006). The representations are learned by making two objects augmented from the same data point close in the vector space, and objects from others as distant as possible. CERT (Fang et al. 2020) adopted an instance-level data augmentation technique (i.e., back-translation). ConSERT (Yan et al. 2021) and SimCSE (Gao, Yao, and Chen 2021) proposed directly adding noise to inner representations of BERT to construct positive pairs, leading to better performance and higher computation efficiency. In this paper, we develop a novel contrastive framework to learn discrete entity states for text generation.

3 Methodology

3.1 Task Definition and Model Overview

Our task is as follows: given a short input such as a beginning $X = (x_1, x_2, \cdots, x_M)$, the model should generate a coherent multi-sentence text $Y = (y_1, y_2, \cdots, y_L)$ (each x_i or y_i is a token). We notice that entity words such as character names usually consist of rare tokens, and the same entity may have different mentions throughout a text (e.g., "Bruce Wayne," "Bruce" and "Wayne"). To better track entity states, we use a two-stage generation framework (Hermann et al. 2015), which first generates a coarse text with all mentions of each entity replaced by a unique placeholder token such as "$\langle$e0$\rangle$, $\langle$e1$\rangle$, $\cdots$," and then generates the mention for each placeholder. We denote the coarse version of the target text with placeholders as $Y^e = (y_1^e, y_2^e, \cdots, y_T^e)$.

[1]The codes are available at https://github.com/thu-coai/ERIC.

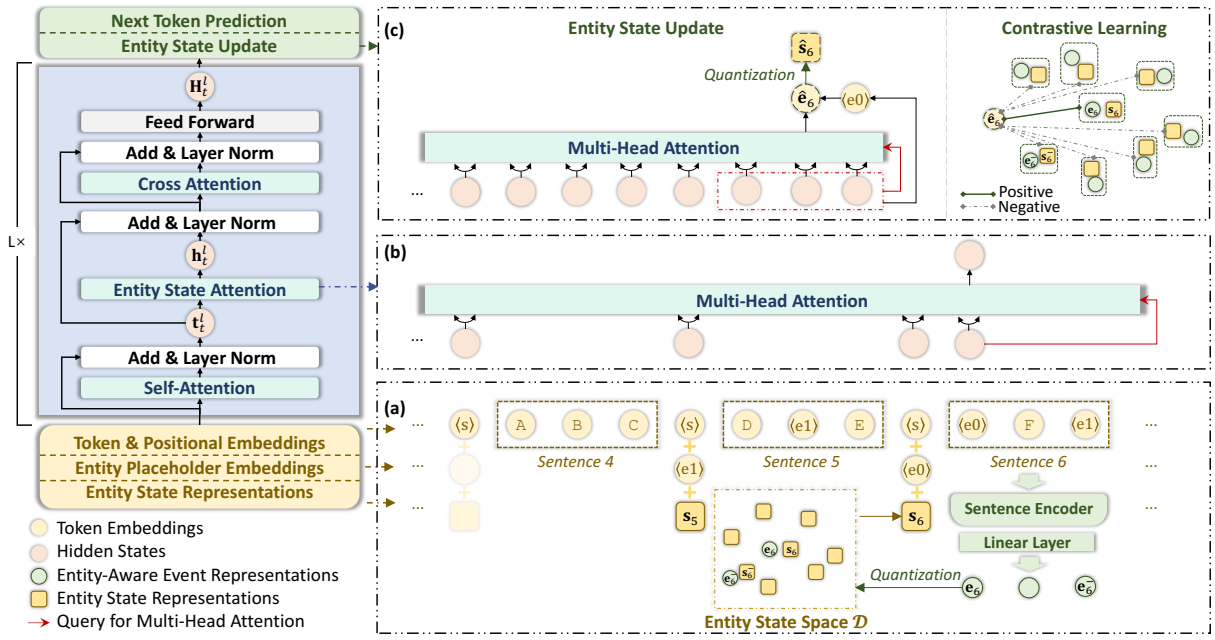

Figure 1: Model overview of the ERIC decoder, which (a) adds a special token ⟨s⟩ before each sentence for incorporating state representations of the following mentioned entities, (b) inserts an entity state attention layer into each Transformer block that narrows the attention scope to only the prefix entity states, and (c) learns an entity state update module with a contrastive objective. We omit the input encoder of ERIC for simplicity.

To generate Y^e from X, generative models such as BART are commonly optimized to minimize the negative log-likelihood $\mathcal{L}_{\text{LM}}$ of Y^e as follows:

$$\mathcal{L}_{\text{LM}} = -\sum_{t=1}^{T} \log P(y_t^e | y_{<t}^e, X). \tag{1}$$

Further, ERIC conducts entity state updates and sentence realization in an auto-regressive manner. At the end of each sentence, we incorporate the state vector of the entity mentioned in the following sentence to the decoder (§ 3.2), which then guides the subsequent generation with an entity state attention layer inserted into each decoder block (§ 3.3). We design a contrastive objective for learning to predict the state of the following mentioned entity conditioned on prefix decoder outputs (§ 3.4). Although we only consider sentences as segments in this paper, our algorithm can easily be extended to other syntactic levels such as paraphrases. Figure 1 shows the model overview of the first training stage. In the second stage, we train another standard Transformer model to generate mentions for anonymous entities (§ 3.5).

3.2 Incorporating Entity States

Suppose that Y^e consists of N sentences, denoted from Y_1^e to Y_N^e (e.g., ABC in Figure 1). We insert a special token ⟨s⟩ at the beginning of each sentence Y_n^e ($n = 1, 2 \cdots, N$) (Guan et al. 2021), which is designed to incorporate entity state representations to guide the realization of Y_n^e. Formally, we set the decoder input $\mathbf{H}_t^0$ as follows:

$$\mathbf{H}_t^0 = \begin{cases} \mathbf{E}_t + \mathbf{E}(p_n) + \mathbf{s}_n, & \text{if } y_t^e = \langle \text{s} \rangle |_n \\ \mathbf{E}_t, & \text{otherwise} \end{cases} \tag{2}$$

$$\mathbf{E}_t = \mathbf{E}(y_t^e) + \mathbf{P}(y_t^e), \tag{3}$$

where $\mathbf{E}_t$ is the sum of the token and positional embeddings of the t-th token y_t^e, $\langle \text{s} \rangle |_n$ means the n-th special token, p_n is the placeholder token of the entity mentioned in Y_n^e, $\mathbf{E}(p_n)$ and $\mathbf{s}_n$ are the token embedding and state vector of p_n, respectively. When Y_n does not mention any entities (e.g., sentence 4 in Figure 1), we set both $\mathbf{E}(p_n)$ and $\mathbf{s}_n$ to zero vectors. While Y_n contains multiple placeholders (e.g., sentence 6 in Figure 1), we randomly sample one as p_n. Our algorithm also adapts to multiple entities, which is left to future work.

We pre-define a discrete entity state space $\mathcal{D} \in \mathbb{R}^{K \times D}$, which consists of K entity states with each state represented as a normalized D-dimensional vector $\mathbf{d}_i$ ($i = 1, 2 \cdots, K$). As shown in Figure 1, we pass a sentence Y_n^e through an external bidirectional sentence encoder, and map the hidden output at the position of p_n (e.g., ⟨e0⟩ in Figure 1) to D dimensions using a linear layer and normalize it as the entity-aware event representation, denoted as $\mathbf{e}_n$. We then derive $\mathbf{s}_n$, the state vector of p_n, through the nearest neighbour look-up from $\mathcal{D}$:

$$\mathbf{s}_n = \mathbf{d}_k, \text{where } k = \underset{\mathbf{d}_j \in \mathcal{D}}{\text{argmin}}\, \mathbf{d}_j \cdot \mathbf{e}_n, \tag{4}$$

where we have normalized both $\mathbf{d}_j$ and $\mathbf{e}_n$ onto the unit hyper-sphere. We use the straight-through trick (Bengio, Léonard, and Courville 2013) to allow the gradient to back-

propagate to the sentence encoder, i.e., directly copying the gradient of $\mathbf{s}_n$ to $\mathbf{e}_n$ (Van Den Oord, Vinyals et al. 2017).

3.3 Entity State Attention Layer

In order to explicitly exert entity states on text generation, we insert an additional entity state attention layer between the vanilla self-attention layer and cross-attention layer in each decoder block. Assuming that the decoder consists of L blocks, let $\mathbf{t}_t^l$ denote the layer-normalized output of the self-attention layer in the l-th block at the t-th position ($t = 1, 2, \cdots, T,\ l = 1, 2, \cdots, L$). The entity state attention layer allows $\mathbf{t}_t^l$ to attend to only those hidden states corresponding to prefix entity states:

$$\mathbf{h}_t^l = \mathrm{A}(\mathrm{Q} = \mathbf{t}_t^l, \mathrm{K/V} = \{\mathbf{t}_{t' \leqslant t}^l | y_{t'}^e = \langle \mathtt{s} \rangle\}), \tag{5}$$

where $\mathbf{h}_t^l$ is the output hidden state of the state attention layer, $\mathrm{A}(\cdot)$ means the multi-head attention mechanism (Vaswani et al. 2017), Q, K and V are the corresponding query, key and value vectors. In this way, ERIC predicts the next tokens with the guidance of entity states using an individual attention network besides the standard self-attention, enhancing its ability to model dependencies between sentence realization and entity states.

3.4 Entity States Learning

ERIC uses entity states derived from the following golden sentences for training. We add an entity state update module on top of the decoder to learn to predict entity states based on prefix information using a contrastive framework, which will be used to guide generation in the inference stage.

The entity state update module consists of two key components, which are used to predict the following mentioned entity $\hat{p}_n$ and its state $\hat{\mathbf{s}}_n$ ($n = 1, 2, \cdots, N$), respectively. We adopt a linear layer to predict the distribution of $\hat{p}_n$ over the vocabulary of all placeholders conditioned on the prefix, and minimize the prediction loss $\mathcal{L}_{\mathrm{Ent}}$ as follows:

$$\mathcal{L}_{\mathrm{Ent}} = -\sum_{n=1}^{N} \log P(\hat{p}_n = p_n), \tag{6}$$

$$P(\hat{p}_n) = \mathrm{softmax}(\boldsymbol{W}_p \mathbf{q}_{n-1} + \boldsymbol{b}_p), \tag{7}$$

$$\mathbf{q}_{n-1} = \mathrm{MeanPool}(\{\mathbf{H}_t^L\}_{n-1}), \tag{8}$$

where $\mathbf{q}_{n-1}$ is a context summary vector derived by applying mean-pooling on $\{\mathbf{H}_t^L\}_{n-1}$, i.e., the set of hidden outputs of the $(n-1)$-th sentence, $\boldsymbol{W}_p$ and $\boldsymbol{b}_p$ are trainable parameters. We add a special token $\langle \mathtt{none} \rangle$ into the placeholder vocabulary with a constant zero embedding to indicate that no entities will be mentioned in the following sentence. We decide $\hat{p}_n$ by taking a sample from $P(\hat{p}_n)$, and predict its state vector $\hat{\mathbf{s}}_n$ as follows:

$$\hat{\mathbf{s}}_n = \mathbf{d}_k, \text{where } k = \underset{\mathbf{d}_j \in \mathcal{D}}{\operatorname{argmin}}\ \mathbf{d}_j \cdot \hat{\mathbf{e}}_n. \tag{9}$$

$$\hat{\mathbf{e}}_n = \mathrm{Normalize}\big(\mathrm{A}(\mathrm{Q} = \mathbf{q}_{n-1} + \mathbf{E}(\hat{p}_n), \quad \mathrm{K/V} = \{\mathbf{H}_t^L\}_{1:n-1})\big), \tag{10}$$

where $\hat{\mathbf{e}}_n$ is the continuous state representation of $\hat{p}_n$ before quantization, and $\{\mathbf{H}_t^L\}_{1:n-1}$ is the set of hidden states of the first $n-1$ sentences. To learn the state representation, we design a contrastive framework to draw $\hat{\mathbf{e}}_n$ close to the following event representation $\mathbf{e}_n$ and keep it away from others derived from different sentences or corresponding to different entities in the same mini-batch (e.g., $\mathbf{e}_6^-$ in Figure 1). To back-propagate gradients to learn the state space $\mathcal{D}$, we set each positive or negative to a joint representation that combines the event representation $\mathbf{e}_n$ and its nearest state vector $\mathbf{s}_n$ in $\mathcal{D}$. In this way, they can be optimized in the same direction and forced to be distributed uniformly in $\mathcal{D}$, thus gaining better expressiveness. We formulate the contrastive objective $\mathcal{L}_{CL}$ as the following infoNCE loss (Oord, Li, and Vinyals 2018):

$$\mathcal{L}_{\mathrm{CL}} = -\frac{1}{N}\sum_{n=1}^{N} \log \frac{\exp(\hat{\mathbf{e}}_n \cdot \mathbf{c}_n/\tau)}{\sum_{\mathbf{c}_n^* \in C} \exp(\hat{\mathbf{e}}_n \cdot \mathbf{c}_n^*/\tau)}, \tag{11}$$

$$\mathbf{c}_n = \mathrm{Normalize}(\mathbf{e}_n + \mathbf{s}_n), \tag{12}$$

where $\mathbf{c}_n$ is the positive joint representation for $\hat{\mathbf{e}}_n$, C is the set of all joint representations in the mini-batch, and τ is the adjustable temperature.

Once we obtain $\hat{\mathbf{e}}_n$, the corresponding discrete state vector $\hat{\mathbf{s}}_n$ can be obtained by quantizing $\hat{\mathbf{e}}_n$ to $\mathcal{D}$ using the nearest neighbour look-up as shown in Eq. 9. And $\hat{p}_n$ and $\hat{\mathbf{s}}_n$ will be taken as input in Eq. 2 to guide the generation in the inference stage. In summary, we train the input encoder, the sentence encoder and the decoder jointly with the following overall loss function:

$$\mathcal{L} = \mathcal{L}_{\mathrm{LM}} + \lambda_1 \mathcal{L}_{\mathrm{Ent}} + \lambda_2 \mathcal{L}_{\mathrm{CL}}, \tag{13}$$

where λ_1 and λ_2 are adjustable sale factors.

3.5 Entity Mention Generation

This stage requires generating superficial entity mentions for placeholder tokens in Y^e. We only use a standard Transformer model for this stage since it is not the main focus of this work. Other training techniques can be also easily applied (Fan, Lewis, and Dauphin 2019). Formally, given X and Y^e, the model should generate a sequence of pairs of a placeholder token followed by its corresponding superficial word in the order that they are mentioned. For example, for the text *"⟨e0⟩ and his girlfriend ⟨e1⟩ take a romantic vacation to a cabin. While in the cabin, ⟨e0⟩ ⋯,"* the golden output is *"⟨e0⟩Ash Williams⟨e1⟩Linda⟨e0⟩Ash ⋯."* After obtaining the entity mentions, we can insert them back into Y^e to get the whole text Y.

4 Experiments

4.1 Datasets

We evaluate ERIC on two English narrative datasets, Wikiplots[2] and CNN News (Hermann et al. 2015). Wikiplots collected story plots from Wikipedia. We use the official split of Wikiplots. We follow Ji and Huang (2021) to split each sentence into sequential elementary discourse units and regard them as sentences for experiments on Wikiplots.

[2] www.github.com/markriedl/Wikiplots

CNN News consists of online newspaper articles collected from CNN. We process the dataset using the script provided by Tan et al. (2021) and split the dataset randomly by 18:1:1 for training/validation/testing. For both datasets, we take the first sentence as input to generate the rest. We remove those texts whose outputs contain less than five sentences, and truncate each output to at most fifteen sentences. We use spaCy[3] to identify people and organization names in a text as entity mentions. If the string of an entity mention is included in another, we replace them with the same placeholder. There are about 98% of examples and 50% of sentences that mention at least one entity on both datasets. More statistics are shown in Table 2, where we count tokens using the NLTK tokenizer (Loper and Bird 2002). More details are shown in Appendix A.1.

Data	Wikiplots			CNN News		
	Train	Val	Test	Train	Val	Test
Num	76,826	4,327	4,324	82,836	4,602	4,602
Len.	237	237	237	346	345	345
#Sen.	12.6	12.7	12.6	14.3	14.2	14.3
#Ent.	6.8	6.7	6.7	7.7	7.6	7.7

Table 2: *Num*ber of examples, average output lengths (*Len.*), numbers of sentences (*Sen.*) and distinct entities (*Ent.*) for *train*ing, *val*idation and *test*ing, respectively.

4.2 Implementation

Our algorithm adapts to all generative models with autoregressive decoders. Due to limited computational resources, we use $\text{BART}_{\text{Base}}$'s pretrained checkpoint for initialization. The sentence encoder is initialized using the pretrained parameters of the $\text{BART}_{\text{Base}}$ encoder. We set the number of discrete entity states in $\mathcal{D}$ to 512, the dimension of state vectors to 128, the maximum number of distinct entities in a text to 100, τ in Eq. 12 to 0.1, and λ_1/λ_2 in Eq. 13 to 1/1. These settings lead to 3% more parameters of ERIC than $\text{BART}_{\text{Base}}$[4]. For both stages, we set the batch size to 12, the maximum sequence length to 512, and the learning rate to 1e-4. We decide the hyper-parameters based on the performance on the validation set.

During inference, we use top-p sampling (Holtzman et al. 2020) with $p = 0.9$ for both generation stages. In the second stage, when the model fails to generate a mention word for a certain placeholder, we complete the output of the first stage using the last word generated for this placeholder if it has been mentioned before (about 2.5% of cases), or a random name otherwise (about 1.1% of cases).

4.3 Baselines

We compared ERIC with the following models: **(1) Seq2Seq:** It has the same architecture as $\text{BART}_{\text{Base}}$ but is initialized randomly. **(2) BART:** It is fine-tuned on the downstream datasets with the standard language modeling objective. **(3) PlanAhead:** It first plans a keyword distribution and then combines the planned distribution with the language model prediction using a gated mechanism (Kang and Hovy 2020). We use $\text{BART}_{\text{Base}}$ as the backbone model and add additional parameters for planning and distribution combination. **(4) HINT:** It incorporates high-level sentence representations into $\text{BART}_{\text{Base}}$ and uses sentence similarity prediction and sentence order discrimination to learn these representations (Guan et al. 2021). **(5) DISCODVT:** It extends $\text{BART}_{\text{Base}}$ to represent the discourse structure using a sequence of latent codes with learnable embeddings (Ji and Huang 2021). We do not limit the minimum length of the latent code sequence like in the original paper. **(6) SimCTG:** It adds a contrastive objective that tries to distribute the hidden outputs of $\text{BART}_{\text{Base}}$ uniformly in the representation space (Su et al. 2022).

Besides the above baselines, we conduct ablation tests on Wikiplots by removing the proposed components respectively. For fair comparison, we insert the special token $\langle \text{s} \rangle$ before each sentence for all baselines. During the evaluation, we remove all special tokens from the generated texts.

4.4 Automatic Evaluation

Evaluation Metrics We do not use perplexity for evaluation since the two-stage generation paradigm of ERIC makes it intractable to assess the text probability. We use the following automatic metrics: **(1) BLEU (B-n)**: It evaluates the n-gram overlap between generated and human-written texts (Papineni et al. 2002), $n = 1, 2$. **(2) MS-Jaccard (MSJ-n)**: It measures the similarity between two n-gram distributions of generated and human-written texts using the Jaccard Index between two multi-sets of n-grams (Alihosseini, Montahaei, and Baghshah 2019), $n = 1, 2$. **(3) MAUVE**: It measures the similarity between two text distributions of generated and human-written texts (Pillutla et al. 2021), where text representations are derived from $\text{GPT2}_{\text{Base}}$. **(4) Token Repetition (R-n)**: It measures the repetition of generated texts by calculating the fraction of the identical token that occurs in the previous n tokens (Welleck et al. 2020), $n = 16, 32, 64$. **(5) Distinct (D-n)**: It measures the generation diversity using the ratio of distinct n-grams to all generated n-grams (Li et al. 2016), $n = 3, 4$. **(6) Zipf Coefficient (Zipf):** It computes the unigram rank-frequency scale factor in generated texts (Holtzman et al. 2020). A value closer to 1 indicates that the generated texts are closer to real-world texts in unigram distribution. Moreover, we also report the average number of generated tokens, denoted as **Len**.

Results Table 3 shows the results on two datasets. The higher BLEU scores of ERIC indicate that it can generate more n-gram overlaps with reference texts than baselines. On the whole, the generation distribution of ERIC is more similar to the ground truth in terms of both n-grams and machine-derived text representations, as shown by the higher MSJ and MAUVE scores. Furthermore, the texts generated by ERIC suffer from less repetition with lower token repetition ratios in various ranges and have better diversity.

[3] https://spacy.io/usage/linguistic-features\#named-entities

[4] We do not count the parameters of the sentence encoder since it is not used during inference.

Models	B-1↑	B-2↑	MSJ-1↑	MSJ-2↑	MAUVE↑	R-16↓	R-32↓	R-64↓	D-3↑	D-4↑	Zipf	Len
Dataset: Wikiplots												
Seq2Seq	26.33	10.28	56.87	39.94	74.56	20.02	32.99	41.61	68.30	89.79	1.26	199
BART	28.33	11.66	58.96	41.16	76.88	17.82	30.74	39.47	70.26	90.51	1.19	201
PlanAhead	27.52	11.32	58.86	41.31	75.98	17.40	30.27	38.90	71.84	91.27	1.17	193
HINT	29.81	12.22	61.17	42.33	80.16	17.20	30.12	39.00	71.73	91.12	1.14	209
DISCODVT	28.76	11.76	60.89	42.29	78.01	17.07	29.81	38.56	73.22	91.76	1.13	204
SimCTG	29.08	11.90	59.81	41.53	77.15	17.68	30.59	39.40	70.44	90.52	1.18	206
ERIC	**31.81**	**12.90**	**63.93**	**42.94**	**87.88**	**16.22**	**28.12**	**37.06**	**75.94**	**93.02**	**1.11**	236
w/o SA	30.12	12.26	62.81	42.84	83.29	16.75	28.56	37.42	75.41	92.85	1.12	221
w/o $\mathbf{s}_n$	28.84	11.70	59.16	40.84	83.18	16.62	28.90	37.43	72.42	91.36	1.15	206
w/o $\mathbf{s}_n$& p_n	28.63	11.59	58.74	40.43	73.01	16.34	28.86	37.93	72.97	91.54	1.12	203
Truth	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*13.63*	*25.30*	*34.75*	*85.80*	*97.10*	*0.96*	*237*
Dataset: CNN News												
Seq2Seq	31.99	14.42	68.25	47.68	85.20	16.06	27.40	36.88	74.02	91.42	1.11	271
BART	32.18	14.66	68.20	47.37	87.25	14.63	25.74	35.34	77.05	92.64	1.08	267
PlanAhead	32.23	14.64	67.05	47.11	28.95	14.55	25.70	35.48	76.53	92.12	1.08	259
HINT	33.07	15.18	69.29	47.74	86.15	14.57	25.69	35.19	77.16	92.61	1.07	275
DISCODVT	32.53	14.92	68.63	47.64	83.43	14.37	25.50	35.09	77.47	92.75	1.07	267
SimCTG	32.50	14.79	68.91	47.75	86.65	14.37	25.56	35.11	76.92	92.48	1.07	272
ERIC	**33.83**	**15.28**	**70.47**	**48.09**	**91.02**	**14.29**	**25.16**	**34.81**	**78.16**	**93.01**	**1.06**	282
Truth	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*12.28*	*22.39*	*32.22*	*82.89*	*94.61*	*1.00*	*345*

Table 3: Automatic evaluation results. ↓ / ↑ means the lower/higher the better. The best performance is highlighted in bold, and the second is underlined. ERIC w/o SA means removing the entity state attention layer. ERIC w/o $\mathbf{s}_n$ means removing entity state representations in the decoder input along with the contrastive learning framework. ERIC w/o $\mathbf{s}_n$ & p_n means further remove the next mentioned entity prediction module.

ERIC also achieves better modeling of long-tail tokens (e.g., entity mentions), with a Zipf score closer to 1. The superiority of ERIC on both datasets proves its generalization for text generation with different lengths and domains. Additionally, we observe that the Wikiplots dataset has more long-tail tokens than CNN News with a lower Zipf score, which may account for the higher superiority of ERIC on Wikiplots since more low-frequency entity mentions may make it harder for baselines to model the entity coherence implicitly.

For ablation tests, the entity state attention layer helps the decoder exploit entity state representations better, thereby improving performance on all metrics and particularly reducing short-range repetition. When removing $\mathbf{s}_n$ or both $\mathbf{s}_n$ and p_n, the BLEU and MSJ scores drop to the level of BART, suggesting the importance of tracking entity states. We also notice that they still have surprisingly less repetition than all baselines. Manual inspection finds that they tend to generate fewer coordinate combinations of identical entity names. For example, there are 14.8% and 13.0% of texts generated by BART and ERIC w/o $\mathbf{s}_n$&p_n, respectively, that contain strings of the form *"W and W"* (*"W"* is a unigram). The phenomenon shows that the two-stage generation paradigm may make it easier to learn entity mention patterns. Significantly, ERIC further surpasses the two ablation models with less repetition thanks to the modeling of entity state transitions which integrates high-level dependencies between events attached to involved entities.

Entity Coherence Modeling It is necessary to investigate whether tracking entity states helps better capture the entity coherence. To this end, on the test set of Wikiplots with masked entity mentions, we replace the first entity placeholder in each sentence in order with another that has been mentioned before randomly (with the prefix not perturbed). Figure 3 plots the accuracy that a model gives a lower probability to the perturbed sentence than the original one along with the prefix. We calculate text probabilities using the following mentioned entities (i.e., p_n) as input for all models, and using the entity states predicted by the model (i.e., $\hat{\mathbf{s}}_n$) as input for ERIC and randomly sampled entity states for ERIC (Rand). We use the models after the first-stage training for this experiment.

We observe that ERIC outperforms ERIC w/o $\mathbf{s}_n$ significantly ($p < 0.01$), especially for the first several sentences. And its performance drops substantially when using random entity states, suggesting that ERIC has a better ability to model entity coherence with the guidance of meaningful entity states. For example, when a placeholder $\langle \mathrm{e1} \rangle$ is replaced to $\langle \mathrm{e0} \rangle$, the event that is attached to $\langle \mathrm{e1} \rangle$ originally may disagree with the state of $\langle \mathrm{e0} \rangle$. In this case, it is easier for ERIC to capture such incoherence issues by tracking their internal states. Appendix C shows more analysis.

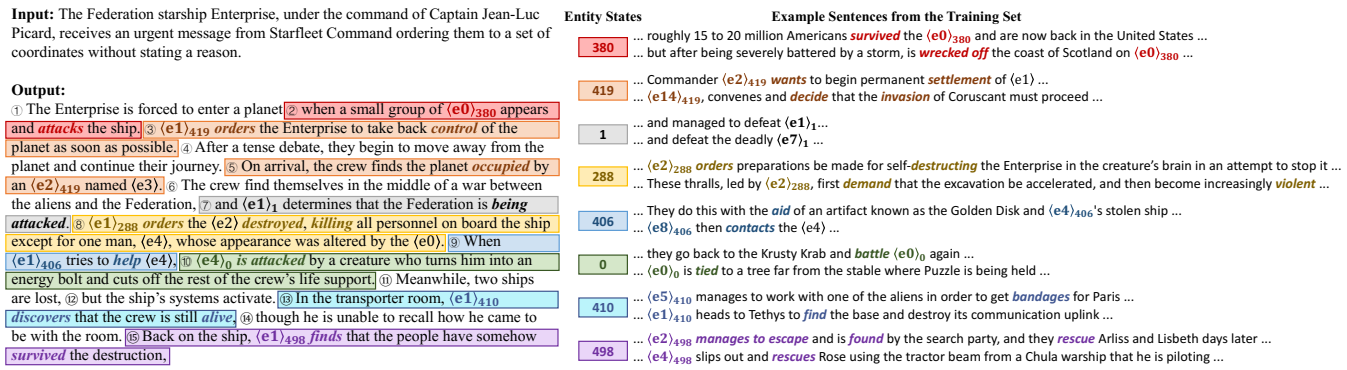

Figure 2: A case generated by ERIC on Wikiplots (Left) and example sentences from the training set where some entity has the corresponding state (Right). The subscript under each entity placeholder token (e.g., $\langle e0 \rangle_{380}$) denotes its state ID. We highlight semantically correlated keywords between the generated case and example sentences in *italic* type. For each entity state, two example sentences are manually selected from top ten sentences with the closest event representations to the state vector.

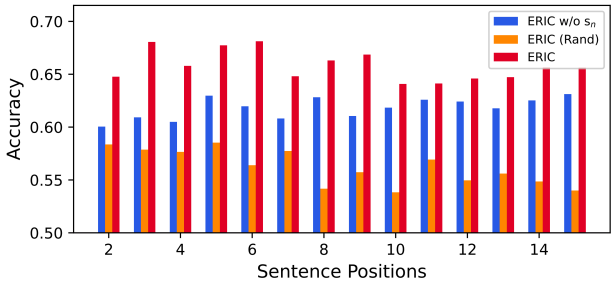

Figure 3: Accuracy of different models varying with positions of perturbed sentences on Wikiplots.

ERIC vs.	Informativeness	Coherence	Issues
BART	55 / 33 / 12	57 / 35 / 8	61/ 87
HINT	58 / 31 / 11	56 / 37 / 8	68/ 81
DISCODVT	57/ 32 / 11	54 / 34 / 12	56/ 83

Table 4: Manual evaluation results on Wikiplots. Left: Percentages (%) of *win* / *lose* / *tie* when comparing ERIC with a baseline. Right: Percentages (%) of texts that suffer from entity issues (ERIC/*the other*). Appendix D shows the kappa scores (Fleiss and Joseph 1971) and significance test results.

4.5 Manual Evaluation

We conduct pair-wise comparisons between ERIC and three strong baselines including BART, HINT and DISCODVT, with each pair of models compared conditioned on 200 inputs randomly sampled from the test set of Wikiplots. We hired three workers on Amazon Mechanical Turk (AMT) to give a preference (win, lose, or tie) in terms of informativeness (interesting, diverse and rich details) and coherence (reasonable inter-sentence dependencies, e.g., causal and temporal relationships). We also asked workers to annotate whether each text exhibits entity issues such as attaching conflicting or repetitive events to an entity. We use majority voting to make final decisions among the workers. The annotation instruction is shown in Appendix D.

As shown in Table 4, all results show fair or better inter-annotator agreement ($\kappa \geqslant 0.2$). ERIC can generate more abundant details with more coherent plots significantly ($p < 0.01$), and suffers from about 20% fewer entity issues than baselines by tracking entity states explicitly.

4.6 Case Study

Figure 2 shows a case to investigate the correspondence between entity states and texts. We conclude that: **(1) The entity states learn meaningful correspondence to specific events.** As shown in the right part of Figure 2, *state 419* relates to "occupying some regions," *state 380* means the entity may be an "attacker" while *state 1* stands for "being defeated." **(2) The entity states effectively guide text generation.** For instance, $\langle e0 \rangle_{380}$ in the second sentence "attacks" the ship. Both $\langle e1 \rangle_{419}$ in the third sentence and $\langle e2 \rangle_{419}$ in the fifth sentence intend to "occupy" the planet. **(3) Tracking entity states helps maintain long-range coherence for text generation.** For example, the protagonist $\langle e1 \rangle$ behaves with reasonable state transitions throughout the text, forming a coherent plot. We show the generation results of baselines in Appendix E.

5 Conclusion

We present the first study to track entity states in large pretraining models for narrative generation. The proposed model ERIC dynamically updates entity states at the sentence level, where each state associates with a cluster of events that can be attached to the entity. Then these states serve to guide the subsequent generation with an additional entity state attention layer. We design a contrastive framework to learn entity-aware event representations and discrete state vectors jointly. ERIC surpasses strong baselines in automatic and manual evaluation for story and news generation. Further analysis shows that ERIC can better capture entity coherence and generate more coherent and informative texts with the guidance of meaningful state representations.

Acknowledgements

This work was supported by the National Science Foundation for Distinguished Young Scholars (with No. 62125604) and the NSFC projects (Key project with No. 61936010 and regular project with No. 61876096). This work was also supported by the Guoqiang Institute of Tsinghua University, with Grant No. 2019GQG1 and 2020GQG0005. This work was also sponsored by Tsinghua-Toyota Joint Research Fund.

References

Alihosseini, D.; Montahaei, E.; and Baghshah, M. S. 2019. Jointly measuring diversity and quality in text generation models. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, 90–98.

Ammanabrolu, P.; Cheung, W.; Broniec, W.; and Riedl, M. O. 2021. Automated Storytelling via Causal, Commonsense Plot Ordering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 5859–5867.

Bengio, Y.; Léonard, N.; and Courville, A. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Clark, E.; Ji, Y.; and Smith, N. A. 2018. 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)*, 2250–2260.

Dai, Z.; Lai, G.; Yang, Y.; and Le, Q. 2020. Funnel-Transformer: Filtering out Sequential Redundancy for Efficient Language Processing. In *NeurIPS*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186.

Fan, A.; Lewis, M.; and Dauphin, Y. 2019. Strategies for Structuring Story Generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2650–2660. Florence, Italy: Association for Computational Linguistics.

Fang, H.; Wang, S.; Zhou, M.; Ding, J.; and Xie, P. 2020. Cert: Contrastive self-supervised learning for language understanding. *arXiv preprint arXiv:2005.12766*.

Fleiss; and Joseph, L. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5): 378–382.

Gao, T.; Yao, X.; and Chen, D. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 6894–6910. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics.

Grosz, B. J.; Joshi, A. K.; and Weinstein, S. 1995. Centering: A framework for modelling the local coherence of discourse.

Guan, J.; Huang, F.; Zhao, Z.; Zhu, X.; and Huang, M. 2020. A knowledge-enhanced pretraining model for commonsense story generation. *Transactions of the Association for Computational Linguistics*, 8: 93–108.

Guan, J.; Mao, X.; Fan, C.; Liu, Z.; Ding, W.; and Huang, M. 2021. Long Text Generation by Modeling Sentence-Level and Discourse-Level Coherence. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 6379–6393. Online: Association for Computational Linguistics.

Guo, M.; Yang, Y.; Stevens, K.; Cer, D.; Ge, H.; Sung, Y.-h.; Strope, B.; and Kurzweil, R. 2019. Hierarchical Document Encoder for Parallel Corpus Mining. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, 64–72. Florence, Italy: Association for Computational Linguistics.

Hadsell, R.; Chopra, S.; and LeCun, Y. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, 1735–1742. IEEE.

Henaff, M.; Weston, J.; Szlam, A.; Bordes, A.; and LeCun, Y. 2017. Tracking the World State with Recurrent Entity Networks. In *ICLR (Poster)*.

Hermann, K. M.; Kocisky, T.; Grefenstette, E.; Espeholt, L.; Kay, W.; Suleyman, M.; and Blunsom, P. 2015. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28.

Holtzman, A.; Buys, J.; Du, L.; Forbes, M.; and Choi, Y. 2020. The Curious Case of Neural Text Degeneration. In *International Conference on Learning Representations*.

Hu, Z.; Chan, H. P.; Liu, J.; Xiao, X.; Wu, H.; and Huang, L. 2022. PLANET: Dynamic Content Planning in Autoregressive Transformers for Long-form Text Generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2288–2305. Dublin, Ireland: Association for Computational Linguistics.

Ji, H.; and Huang, M. 2021. DiscoDVT: Generating Long Text with Discourse-Aware Discrete Variational Transformer. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 4208–4224.

Ji, Y.; Tan, C.; Martschat, S.; Choi, Y.; and Smith, N. A. 2017. Dynamic Entity Representations in Neural Language Models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 1830–1839.

Kang, D.; and Hovy, E. 2020. Plan ahead: Self-Supervised Text Planning for Paragraph Completion Task. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 6533–6543. Online: Association for Computational Linguistics.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In Jurafsky, D.; Chai, J.; Schluter, N.; and Tetreault,

J. R., eds., *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, 7871–7880. Association for Computational Linguistics.

Li, J.; Galley, M.; Brockett, C.; Gao, J.; and Dolan, B. 2016. A Diversity-Promoting Objective Function for Neural Conversation Models. In Knight, K.; Nenkova, A.; and Rambow, O., eds., *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, 110–119. The Association for Computational Linguistics.

Li, J.; Luong, M.-T.; and Jurafsky, D. 2015. A Hierarchical Neural Autoencoder for Paragraphs and Documents. 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)*, 1106–1115.

Loper, E.; and Bird, S. 2002. Nltk: The natural language toolkit. *arXiv preprint cs/0205028*.

Miculicich, L.; Ram, D.; Pappas, N.; and Henderson, J. 2018. Document-Level Neural Machine Translation with Hierarchical Attention Networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2947–2954. Brussels, Belgium: Association for Computational Linguistics.

Nawrot, P.; Tworkowski, S.; Tyrolski, M.; Kaiser, Ł.; Wu, Y.; Szegedy, C.; and Michalewski, H. 2021. Hierarchical Transformers Are More Efficient Language Models. *arXiv preprint arXiv:2110.13711*.

Oord, A. v. d.; Li, Y.; and Vinyals, O. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Papalampidi, P.; Cao, K.; and Kocisky, T. 2022. Towards Coherent and Consistent Use of Entities in Narrative Generation. *arXiv preprint arXiv:2202.01709*.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 311–318.

Pillutla, K.; Swayamdipta, S.; Zellers, R.; Thickstun, J.; Welleck, S.; Choi, Y.; and Harchaoui, Z. 2021. MAUVE: Measuring the Gap Between Neural Text and Human Text using Divergence Frontiers. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*, volume 34, 4816–4828. Curran Associates, Inc.

Rashkin, H.; Celikyilmaz, A.; Choi, Y.; and Gao, J. 2020. PlotMachines: Outline-Conditioned Generation with Dynamic Plot State Tracking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 4274–4295.

Ribeiro, M. T.; Wu, T.; Guestrin, C.; and Singh, S. 2020. Beyond Accuracy: Behavioral Testing of NLP Models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 4902–4912. Online: Association for Computational Linguistics.

Su, Y.; Lan, T.; Wang, Y.; Yogatama, D.; Kong, L.; and Collier, N. 2022. A Contrastive Framework for Neural Text Generation. *arXiv preprint arXiv:2202.06417*.

Tan, B.; Yang, Z.; Al-Shedivat, M.; Xing, E.; and Hu, Z. 2021. Progressive Generation of Long Text with Pretrained Language Models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 4313–4324.

Van Den Oord, A.; Vinyals, O.; et al. 2017. Neural discrete representation learning. *Advances in neural information processing systems*, 30.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Welleck, S.; Kulikov, I.; Roller, S.; Dinan, E.; Cho, K.; and Weston, J. 2020. Neural Text Generation With Unlikelihood Training. In *International Conference on Learning Representations*.

Xu, P.; Patwary, M.; Shoeybi, M.; Puri, R.; Fung, P.; Anandkumar, A.; and Catanzaro, B. 2020. MEGATRON-CNTRL: Controllable Story Generation with External Knowledge Using Large-Scale Language Models. In Webber, B.; Cohn, T.; He, Y.; and Liu, Y., eds., *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, 2831–2845. Association for Computational Linguistics.

Yan, Y.; Li, R.; Wang, S.; Zhang, F.; Wu, W.; and Xu, W. 2021. ConSERT: A Contrastive Framework for Self-Supervised Sentence Representation Transfer. 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)*, 5065–5075. Online: Association for Computational Linguistics.

Yang, Z.; Yang, D.; Dyer, C.; He, X.; Smola, A.; and Hovy, E. 2016. Hierarchical Attention Networks for Document Classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1480–1489. San Diego, California: Association for Computational Linguistics.

Yao, L.; Peng, N.; Weischedel, R.; Knight, K.; Zhao, D.; and Yan, R. 2019. Plan-and-write: Towards better automatic storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 7378–7385.

Zhang, X.; Wei, F.; and Zhou, M. 2019. HIBERT: Document Level Pre-training of Hierarchical Bidirectional Transformers for Document Summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 5059–5069.