A Hybrid Model for Globally Coherent Story Generation

Zhai Fangzhou[†], **Vera Demberg**[†,‡], **Pavel Shkadzko**[†], **Wei Shi**[†] and **Asad Sayeed**[*]
[†] Dept. of Language Science and Technology
[‡] Dept. of Mathematics and Computer Science, Saarland University
[*]Dept. of Philosophy, Linguistics and Theory of Science, University of Gothenburg
{fzhai, vera, w.shi}@coli.uni-saarland.de
p.shkadzko@gmail.com, asad.sayeed@gu.se

Abstract

Automatically generating globally coherent stories is a challenging problem. Neural text generation models have been shown to perform well at generating fluent sentences from data, but they usually fail to keep track of the overall coherence of the story after a couple of sentences. Existing work that incorporates a text planning module succeeded in generating recipes and dialogues, but appears quite data-demanding. We propose a novel story generation approach that generates globally coherent stories from a fairly small corpus. The model exploits a symbolic text planning module to produce text plans, thus reducing the demand of data; a neural surface realization module then generates fluent text conditioned on the text plan. Human evaluation showed that our model outperforms various baselines by a wide margin and generates stories which are fluent as well as globally coherent.

1 Introduction

Automatic story generation is the task of automatically determining the content and utilizing proper language to craft stories. One of the most important aspects of these stories is their coherence. The scope of global coherence includes arranging the contents in a plausible order, staying on topic, and creating cohesion through anaphoric expressions, etc.

Traditionally, story generation is performed with symbolic planning systems (see, e.g., Meehan, 1976; Riedl and Young, 2010; Busemann and Horacek, 1998). These systems often follow a hierarchical pipeline: higher level modules perform text planning, determine discourse relations and contents of each sentence; lower level modules account for surface realization accordingly. Although capable of producing impressive, coherent stories, these systems rely heavily on manual knowledge engineering to select actions, characters, etc., properly, therefore generalizing poorly to unseen domains.

Early NLG systems, on the other hand, excel at generating fluent on-topic utterances (see, e.g., Mei et al., 2015; Wen et al., 2015). These models are data-based, and therefore can be applied to new domains if data is available. However, these models struggle to keep track of longer story or dialog history, i.e., they may switch topics, repeat information or say things that are not consistent with sentences generated earlier (see, e.g., Vinyals and Le, 2015; Shang et al., 2015). Extra modelling efforts or knowledge input is required to improve global coherence.

Neural NLG systems that incorporate text planning efforts could improve global coherence, and grant some controllability over the contents, i.e. making it possible to generate stories given specific input of what should happen in the story. (Fan et al., 2019) uses a convolutional seq2seq model to generate a chain of predicate-argument structures from a prompt to sketch a story, and then uses another convolutional seq2seq model to convert the chain of predicate-argument structures to text. The *neural checklist* model (Kiddon et al., 2016) keeps track of the progress of recipe generation with the usage of ingredient words, to generate recipes from a bag of ingredients. If we consider the task of story generation, *events* ("script" events; scripts capture knowledge about "standardized sequences of events about daily activities such as going to a restaurant or visiting a doctor") are also fairly informative for the narration progress. Thus if one could identify events within surface texts, it is, in principle, possible to regard the events as indicators of the narration progress (just like the ingredients in a recipe) and apply the *neural checklist* model.

These requirements are fulfilled by the

Proceedings of the Second Storytelling Workshop, pages 34–45
Florence, Italy, August 1, 2019.

InScript (Modi et al., 2017a) corpus, a small corpus that contains a total of about 1000 stories about everyday events from 10 scenarios. Each story is annotated with script-relevant event types to align them with surface language. However, surprisingly, when we apply the *neural checklist* model on InScript to generate stories, the quality of the generated stories appears poor (see the second item in Table 1 for a sample generation).

We notice two possible reasons for why the *neural checklist* model does not perform well on the task: (1) the scale of InScript is less that 0.5% of the recipe task investigated in (Kiddon et al., 2016), thus the model may not be able to properly fit its complex structure and learn to plan the text; (2) each script event in our data corresponds to multiple possible surface realization options (e.g., both 'I went to the shop' and 'I drove to the supermarket' would be labeled as a go_to_store event), which makes the alignment between an event and the surface text inherently more complicated than that between ingredients and surface text.

To tackle these issues, we propose a new neural story generation model that exploits explicit, symbolic text planning. The model consists of two components: the ***agenda generator*** produces an agenda, a sequence of script events that would later be fleshed out to yield a story, by a neural ***surface realization module***; the neural surface realization module treats the events in the agenda as a latent variable that encodes the progress of story generation, and produces text conditioned on it. The outcome is a system that could be trained with much less data. To our knowledge, this is the first attempt to integrate a completely symbolic text planner with a neural surface realization component, to perform fully interpretable text planning. Human evaluation shows that our system significantly outperforms various baselines in terms of fluency and global coherence.

Our contributions are as follows:

- We develop a story generation model that generates globally coherent stories about daily activities.
- We propose a novel way to combine a neural story generation model with an explicit, symbolic text planning component; furthermore, we show that the design reduces the demand on training data.
- We illustrate the possibility of guiding the direction of story generation by conditioning the generation on a latent intention variable.

In the remainder of this paper, we start with a discussion of related research, and then introduce the InScript corpus. To follow is a detailed introduction of our model and the results from human evaluation. Analysis of the results and some discussions about future directions conclude the paper.

2 Related Work

NLG Conditioned on Latent Intention Variable

Incorporating a latent intention variable in NLG systems yields improved controllability over the content. It is proved effective in data-to-dialogue generation (see, e.g., Yarats and Lewis, 2017; Kiddon et al., 2016). In neural text generation, some domain-specific categories of words are also informative for the progress of generation. Kiddon et al. (2016) developed an end-to-end neural text generation model, which keeps track of the usage of the keywords (e.g. recipe ingredients) with attention mechanism, and conditions surface realization on the usage of these words.

NLG with Explicit Text Planning

Noting that RNN based language models could only account for local coherence, attempts have been made to perform text planning on a higher level (e.g. Jain et al., 2017; Peng et al., 2018). Puduppully et al. (2018) performs content selection and planning with attention based neural networks before surface realization, to generate specifically structured NBA game summaries. Martin et al. (2018) uses sequence to sequence neural network (*event2event*) to generate events (represented by a verb and its most important arguments) corresponding to consecutive sentences. A second sequence to sequence model (*event2sentence*) generates a sentence based on the event. Our method differs from that of Martin et al. (2018), mainly in that (1) we seek to implement text coherence on a document level whereas they mostly focused on consecutive sentence pairs; (2) we do not incorporate explicit sentence segmentation but leave the job to the surface realization component.

3 Data

Our work is based on the `InScript` corpus. We mainly utilized its event annotations and the ***temporal script graphs*** extracted from the corpus.

3.1 The `InScript` Corpus

The `InScript` corpus (Modi et al., 2017a) was designed for the investigation of script knowledge. The corpus includes around 100 stories for each of 10 common daily scenarios. These stories are annotated with event types as well as participant types. For this paper, we only exploit the event type annotations. An example is shown in Figure 1. The average story length is approximately 240 tokens; the corpus includes 238k tokens in total. We use the corpus to train the neural surface realization component.

3.2 The Temporal Script Graphs

Wanzare et al. (2017) compiled the `InScript` event annotations into *temporal script graphs* (see Figure 2). These directed graphs contain information on the typical temporal order of events in a script scenario, which is a crucial aspect of script knowledge. In our method, temporal script graphs are used to generate plausible sequences of events for building the agenda.

4 Our Model

Overview

Our model consists of three modules. Firstly, a symbolic ***agenda generator***, which is responsible for performing text planning. Given a specific scenario (e.g., `baking a cake`), it produces an agenda according its temporal script graph. Secondly, a neural ***surface realization module***, which performs two tasks: (1) it predicts the next word of the story conditioned on the text history and the event that needs to be realized at a specific point in the story; (2) it determines whether the current event has been completely realized so the generation could move to the next event in the agenda. Finally, a ***story generator*** which performs the following. (1) Calls the *agenda generator* to generate an agenda. (2) Creates a seed, a short, plausible beginning, to initialize surface realization, e.g., '*yesterday i went grocery shopping*'. (3) Iteratively calls the surface realization module to perform a beam search (see, e.g., Sutskever et al., 2014) and generate a complete story. (4) Removes occasional (approx. once per thousand tokens) excessive repetitions in the generated story. More precisely, when a word or phrase is repeated at least three times, the third repetition would be deleted. e.g., 'i like the tree very very very much' becomes 'i like the tree very very much'. The generation terminates when the agenda is exhausted and a sentence-terminating punctuation is generated.

4.1 The Agenda Generator

Given a scenario, the *agenda generator* goes through the temporal script graph and samples a path through it. For the example given in Figure 2, the path would start out with "choose recipe" and continue with either "get ingredients" or "buy ingredients", followed by "add ingredients", until the end of the graph is reached. The *agenda generator* also decides whether each event should be realized. In natural stories, narrators usually do not mention all of the events, and this component enables our model to mimic this behavior: the probability of event realization depends on the likelihood of the event given its predecessor $p(e|e')$, which is estimated on the training data using an event bigram model. To avoid excessive discontinuity in the realization, the *agenda generator* is prohibited to skip two consecutive events. The outcome of this process is an agenda, a plausible sequence of events.

Due to its symbolic nature, the *agenda generator* demands no extra training data, which is crucial for reducing the demand of data. Moreover, as the agenda generation is fully transparent and interpretable, we gain fair controllability over the content. For example, dropping a specific event from the agenda would cause the generation to skip it. Actually, it is also possible to use the surface realization module independently and generate a story from an event sequence as input.

4.2 The Neural Surface Realization Module

Our neural surface realization module is a GRU (Cho et al., 2014) language model, modified to enable two additional functionalities. (1) Conditioning the prediction of the successive word on the generation progress. (2) Determining whether the current event has been completely verbalized. If so, the surface realization module shifts its focus one event onward along the agenda and begins to instantiate the next event. See Figure 3 for a conceptual illustration.

(I)$^{(1)}_{\text{P\_bather}}$ [decided]$_{\text{E\_wash}}$ to take a (bath)$^{(2)}_{\text{P\_bath}}$ yesterday afternoon after working out . Once (I)$^{(1)}_{\text{P\_bather}}$ got back home , (I)$^{(1)}_{\text{P\_bather}}$ [walked]$_{\text{E\_enter\_bathroom}}$ to (my)$^{(1)}_{\text{P\_bather}}$ (bathroom)$^{(3)}_{\text{P\_bathroom}}$ and first quickly scrubbed the (bathroom tub)$^{(4)}_{\text{P\_bathtub}}$ by [turning on]$_{\text{E\_turn\_water\_on}}$ the (water)$^{(5)}_{\text{P\_water}}$ and rinsing (it)$^{(4)}_{\text{P\_bathtub}}$ clean with a rag . After (I)$^{(1)}_{\text{P\_bather}}$ finished , (I)$^{(1)}_{\text{P\_bather}}$ [plugged]$_{\text{E\_close\_drain}}$ the (tub)$^{(4)}_{\text{P\_bathtub}}$ and began [filling]$_{\text{E\_fill\_water}}$ (it)$^{(4)}_{\text{P\_bathtub}}$ with warm (water)$^{(5)}_{\text{P\_water}}$ set at about 98 (degrees)$^{(6)}_{\text{P\_temperature}}$.

Figure 1: An excerpt from a story on TAKING A BATH in the InScript corpus taken from Modi et al. (2017b). The referring expressions are in parentheses, and the corresponding discourse referent label is given by the superscript. Referring expressions of the same discourse referent have the same color and superscript number. Script-relevant events are in square brackets and colored in orange. Event types are indicated by the subscripts.

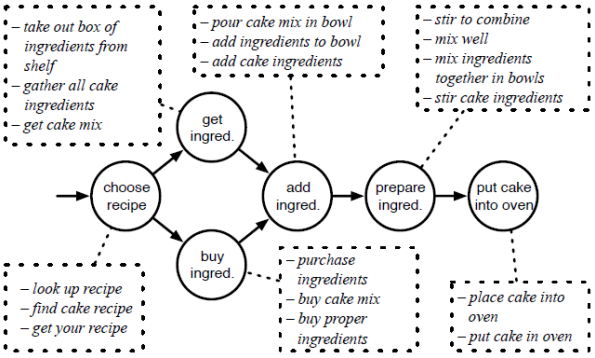

Figure 2: The *Temporal Script Graphs* for the BAKING A CAKE script induced from the InScript corpus, taken from (Wanzare et al., 2017). The nodes are the event clusters whereas the dashed boxes include some possible utterances that correspond to these clusters.

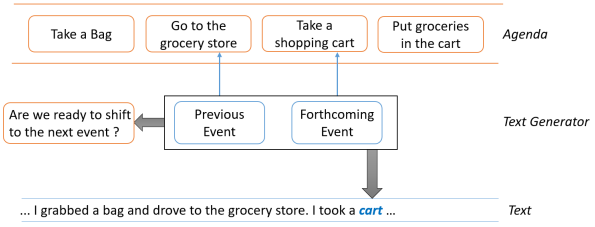

Figure 3: An illustration of the surface realization module. It produces two outputs: a distribution over the vocabulary that predicts the successive word, and a boolean-valued variable that indicates whether the generation should move to the next event.

For the first functionality (see Figure 4 for the model architecture), we condition the prediction of the next word on both the previously instantiated event (the ***preceding event***) and the event that should be verbalized now (the ***forthcoming event***). Intuitively, the surface realization module will be informed with something like '*I have taken a shopping cart, now tell me how to get my groceries*'. We train a dense vector representation for each event in the corpus, which we term ***event vectors***. To condition the surface realization on the events, we grant the generator access to the event vectors e^p_t of the *preceding* event and e^f_t of the *forthcoming* event:

$$d_t = Softmax(D[o_t; e^p_t; e^f_t])$$

here d_t is the output distribution that predicts the successive word; D is an affine transformation; ';' stands for vector concatenation; $o_t = Wh_t$ is the content from the GRU language model where h_t is the GRU cell states, and W is another affine transformation. To further relate the surface realization with the generation progress, we concatenate the event vectors with the embedding of previous word as the input to the GRUs:

$$h_t = GRU([x_{t-1}; e^p_{t-1}; e^f_{t-1}])$$

As a direct consequence, we need to train dense vector representations of all words in the vocabulary. This is quite ambitious, as the corpus is fairly small-scale (about 238k tokens). To alleviate this data sparsity issue, we initialize our word embeddings with Google's pre-trained word2vec vectors[1] (see, e.g., Mikolov et al., 2013). The effectiveness of this domain-adaptation method in language modelling is observed in Zhang et al. (2016). As a side effect, our word embedding dimensionality is fixed at 300.

To determine whether the *forthcoming event* has been instantiated, i.e. whether the model is ready to move onwards, we integrate a binary classifier into the architecture:

$$a_t = Softmax(A[h_t; e^p_t; e^f_t])$$

here A is a projection matrix; a_t is a 2-dimensional vector. If $a^1 > a^0$, the surface realization module decides that the *forthcoming event* has been completely narrated and it should move one event onwards to continue the generation; otherwise, it stick with the current *forthcoming event* to complete its instantiation.

[1] https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit

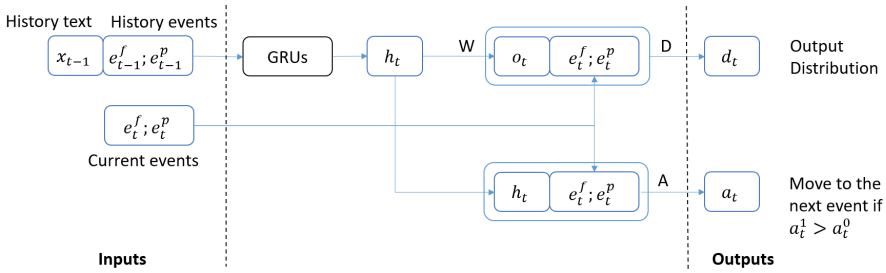

Figure 4: An illustration of the surface realization model architecture. It exploits a multi-task learning framework: it outputs the distribution over the next token d_t, as well as a_t, which determines whether to shift to the next event.

As there are more tokens than events in the corpus, the aforementioned binary classification is biased. The ratio between the categories is about $0.85 : 0.15$. To balance the performance on both categories, we apply greater weight on the loss of the less frequent category. More concretely, we measure a weighted cross-entropy loss on output a_t:

$$L_a(a_t, a_t^*; \gamma) = \begin{cases} -(1-\gamma)\log a_t^0, & a_t^* = (1,0) \\ -\gamma \cdot \log a_t^1, & a_t^* = (0,1) \end{cases}$$

here a_t^* is the ground truth; the weight coefficient γ is a hyper-parameter. All in all, the surface realization module exploits a multi-task learning architecture. The final loss function is

$$L(a_t, a_t^*, d_t, d_t^*; \gamma, \beta) = H(d_t, d_t^*) + \beta \cdot L_a(a_t, a_t^*; \gamma)$$

where $H(\cdot, \cdot)$ denotes the cross entropy between the parameters; d_t^* is the ground truth distribution of the next word (herein: a one-hot vector); β is another hyper-parameter.

It is worth noting that we did not incorporate explicit sentence planning, but completely rely on the surface realization module to perform sentence segmentation. The underlying reason is the absence of a clear correspondence between the agenda events and the sentences: multiple events could appear in a single sentence; likewise, multiple sentences could be devoted to one single event. We count on the surface realization module to punctuate correctly and generate syntactically correct sentences.

5 Experiments

5.1 Experimental Set-up and Optimization

2.5% of the data were randomly selected as the validation set; the rest was kept as the training set. As evaluation will be up to humans instead of any test set metric (see section 5.2), no test set is necessary.

The model was implemented with Python 3.5. The neural network part of the model was implemented with Keras 2.1.2 (Chollet et al., 2015). Optimization was performed with adam optimizer (Kingma and Ba, 2014) with gradient clipping to stabilize the training (see Pascanu et al., 2013). To regularize the model, dropout (Srivastava et al., 2014) was applied to all dense connections, including the explicit dense layers and the fully-connected layers within the GRU cells; besides, we applied early stopping, which monitors the loss function as is defined in section 4.2.

Hyper-parameters are tuned with a two-stage random hyper-parameter search, which is empirically proven more effective than grid search (see Bergstra and Bengio, 2012). On the validation set, the model yields a 0.90 accuracy and a 0.75 $F1$ score on the binary classification task concerning output a (whether to shift to the next event; see section 5.3.1 for some discussion on its consequences) and a 38.9 perplexity for predicting the next word. [2].

5.2 Evaluation

5.2.1 Model Variants

We re-implemented the neural checklist model by Kiddon et al. (2016) as a baseline. We decided not to use Martin et al. (2018) because the sentences it generates are not lexicalized, i.e. they include word categories like `entity.n.01`, which is not directly suitable for human evaluation. Substituting these category labels with surface language is substantially more difficult for our domain than theirs. We also included a human ceiling and sev-

[2] In case of interest in reproducing our result, appendix A provides full details on hyper-parameter tuning; the code is available at https://github.com/arkgithubforyou/story_generation

eral ablated versions of our final model into the human evaluation. To follow is a list of the systems we evaluated.

- **`Human Author`**
  Stories directly taken from the `InScript` corpus. Expected to produce an upper bound of the evaluation.

- **`Full`**
  Our model as is described in section 4.

- **`GRU`**
  A plain GRU language model trained on `InScript`, intended to be a baseline that has no specific global coherence control. Its generations are seeded with scenario-specific beginnings for some relevance. For example, the seed for the `Going Grocery Shopping` script is '*yesterday i went grocery shopping .*'

- **`Neural Checklist`**
  The *neural checklist* model as is described in Kiddon et al. (2016). We applied a post-processing similar to the one described in section 4 to clean up the repetitions.

- **`Random Event Order`**
  A variant of our model with the *agenda generator* ablated. As a substitution, the agendas are now generated by randomly sampling a sequence of events from the set of events corresponding to the respective script. We compare this variant with the `full` model to verify the contribution of the *agenda generator* in implementing global coherence.

For some intuition, see table 1 for sample generations from these systems.

5.2.2 Evaluation Method

Automatic evaluation of text quality, especially its global coherence, is a challenging task (see, e.g. Lapata and Barzilay, 2005; Purdy et al., 2018, for some meaningful attempts though). We also observed poor correlations between a few automatic metrics and the results of human evaluation (see appendix C for more details), and decided that automatic metrics are not suitable for our task. Thus we performed human evaluation through crowdsourcing; it evaluates the following aspects of generated stories.

- **Syntax**
  The syntactical correctness of the sentences.

- **Global Coherence**
  The global coherence of a story with regard to the given script, e.g., `GOING GROCERY SHOPPING`. We evaluate from three aspects: ***Inclusion*** (does the story cover the most necessary steps about the topic?), ***Relevance*** (does the story stay on-topic, and rarely mention anything irrelevant to the topic?), and ***Order*** (does the story describe the activities relevant to the topic in a plausible order?)

- **Agenda Coverage**
  The correspondence between the generated story and the agenda it was fed with. The participants were asked whether each of the agenda items has been realized in the story.

We ask participants five questions per story: for *Agenda Coverage*, participants were asked to check off the agenda items that were mentioned in the story they saw; for the other four aspects, participants were asked to rate on a 1 to 4 scale. The evaluation experiment was implemented with Lingoturk (Pusse et al., 2016); we hired participants and conducted the experiment on Prolific[3]. See appendix B for more details on conducting the experiment of human evaluation.

5.3 Results

5.3.1 Human Evaluation

Table 2 illustrates the results from human evaluation. The `GRU` model, a plain language model without coherence modeling, yields the worst performance on all metrics. The output wildly changes between topics and is incoherent globally; the poor coherence probably also negatively affects human judgments on syntactic correctness. The `neural checklist` saw better performance than plain GRUs, but it failed to include the most necessary steps of the scenario. It seems the model cannot correctly track the progress of the generation, which, as discussed in section 1, we suspect to be a consequence of the limited amount of training data: as its attention-based content-planning cannot make use of the order information and has to learn it from data, the model (and probably also other attention-based models) has a substantially higher demand on training data.

[3] `https://prolific.ac/`

GRU
yesterday i went grocery shopping . i did n't know anything all the items in my cart , so that it was ready to pick up up . i got my items off , and found it to the front of the store . i was on narita shopping cart because i had less than twenty of them . i grabbed my cart and went to the cashier . i paid for the items , then i gave the cashier my money and my bag , checked my bags up , and the other items in my hand bag . i am glad i 'm flying for my plane and go through security .
Neural Checklist
yesterday i wanted to go grocery shopping . when i went to the front of the store , i went to the grocery store and drove to the front of the store . i went to the cashier and drove to the front desk .
Random Event Order
yesterday i went grocery shopping . i grabbed a shopping cart and put the items on the list . i put my items in the cart and drove to the store . i drove to the grocery store and went to the grocery store . i made a list of all the items i would need . i put my items on the cart and headed to the store . when i entered the store , i grabbed a shopping cart and walked inside . i put the cart away in the cart .
Agenda: *evoking→ take shop cart→ put conveyor→ get groceries→ go grocery→ move section→ make list→ check off→ put conveyor→ leave→ enter→ take shop cart→ enter→ return shop cart→ story ends*
Full
yesterday i went grocery shopping . i made a list of my list and drove to the grocery store . when i entered the store , i grabbed a shopping cart and pushed the cart down to the meat aisle . i got all my items , and crossed items on my list . i went to the checkout register and paid for my groceries . i put my groceries in my cart and left .
Agenda: *evoking→make list→go to store→enter→take cart→move along sections→take grocery→check list→go to checkout→pay→pack grocery→leave→ story ends*
Human Author
yesterday i went grocery shopping . i took my grocery list with me , along with some reusable shopping bags . my grocery list has all the items i want to buy on it . i selected a shopping cart from in front of the store , and went inside . i put my reusable bags in the cart . i looked at my list and started in the produce section . i put different vegetables and fruits into my cart . next i wheeled my cart to the cereal aisle and took a box of cereal . i went through the store aisle by aisle and selected my groceries . each aisle is organized by types of food and non-food items . one aisle has dried pasta , canned tomatoes , rice , and sauce . i selected a few boxes of pasta and some rice . another aisle carries plastic wrap , trash bags , and aluminum foil . as i went through the store , i kept looking at my list to see what i needed next . when i added each item to my cart , i crossed it off my list . my last stop was the dairy aisle where i got milk and eggs . when i had all the groceries i wanted , i went to the cash register ans stood in line . when it was my turn , i put each item on the conveyor belt and the cashier scanned each one . a bagger put all of the groceries into my reusable bags . i paid , and then the cashier gave me a receipt . i loaded the bags of groceries into the trunk of my car and drove home .

Table 1: Sample generations by different models on GOING GROCERY SHOPPING. The corresponding seeds are displayed in boldface. **Neural Checklist, Full** used the same agenda, which is given in the table.

	Agenda Coverage**	Syntax	Inclusion	Order	Relevance
human author	86%	0.86	0.91	0.93	0.83
full	**71%**	**0.75**	**0.67**	**0.75**	**0.88**
random event order	50%	0.45	0.46	0.14*	0.71
Neural Checklist	20%	0.54	0.34	0.27	0.53
GRU	n/a	0.33	0.24	0.11*	0.22

*: difference between the pair is not statistically significant due to paired T-test on a significant level $\alpha = 0.05$.

**: answers to the agenda coverage questions yield a Fleiss' kappa of 0.34.

Table 2: Results from human evaluation. Highest scores out of automatic systems are displayed in boldface.

`Human Author` : event *'make a shopping list'* in scenario *'going grocery shopping'*
... next , i used the wipes the store provides at the entrance and wipe off the handle of the shopping cart , and my hands , so i know my hands will stay clean while i choose my food . then ***i took out the shopping list i wrote at home*** and i started . i always start with heavy things ...
`Full Model` : event *'place fertilizer'* in scenario *'planting a tree'*
... yesterday i planted a tree. first , i decided to buy a small apple tree . i got a shovel and drove to the home . i found a perfect spot in my backyard and dug a hole . ***i put*** the soil in the hole and then watered it . ...

Table 3: Examples where instantiations of agenda items failed to be approved by evaluators. Up: event instantiation that was not explicit enough; down: event that was not instantiated due to an error in the output a from the surface realization module.

The `Full` model was able to significantly outperform all other automatic variants and received positive scores for all criteria. It reflects well the events on the agenda and usually includes the most necessary steps of the scripts in a plausible order, which indicates decent global coherence. It even received a higher *relevance* score than `Human Author`. However, this may result from our model often producing shorter stories than the human originals, see section 5.3.2. Its *agenda coverage* score is lower than that of `Human Author`. We detected two sources of these errors: (1) event instantiations are sometimes not recognized as such by the participants, because they are not explicit enough; this is also the reason for why the agenda coverage score for the original human texts is less than 100%. (2) Errors in the event termination judgments of the surface realization module: when the surface realization module wrongly decided that the *forthcoming event* has been instantiated, it would simply skip the event in the generation. See table 3.

`Random Event Order` witnessed a dramatic performance drop compared to `Full`. Its order score is not significantly different from that of the `GRU` baseline. That means, particularly, our *agenda generator* was crucial for and capable of performing reliable text planning and incorporating global coherence. It retained high relevance score (i.e., it still stays on-topic), as the agendas it use were still about the respective scenarios. However, unexpectedly, the *inclusion* score and *syntax* score also saw a sharp drop. For that we noticed two possible origins. Firstly, it might result from a systematic error of human evaluation – the stories produced by the random model, violating global coherence, are in general messy and would make the assessment cognitively difficult. Thus they are likely to receive lower scores. Secondly, our surface realization is conditioned on a 'previous event / forthcoming event' pair, therefore, for less plausible agendas (e.g., one produced by a random agenda generator), the corresponding pairs would appear less frequently in our small-scale corpus, thus suffering more from data sparsity issues and affect the quality of surface realization.

5.3.2 Qualitative Analysis

Most noticeably, the stories our model generates are less elaborative than the corpus stories. From the samples in table 1, we could see that the story from the `full model` is much shorter than the one taken from the corpus. It turns out that our system often chose not to elaborate on an event: a genuine human would occasionally list what she bought from the grocery store, like vegetables, fruits, pasta, rice; whereas our system would only say '*i got my items*'. The most important reason behind this is that these elaborations are sparse, thus whenever our system sees '*i got my items*' in the history, it will decide that the event `take items` is already instantiated, and move onwards to the next event. Another reason for generating shorter stories is our *agenda generator* cannot correctly reproduce some real-world scenarios where the events 'cycle'. For example, the event chain corresponding to a story about `taking a bus` could occasionally look like $borad \rightarrow ride \rightarrow exit \rightarrow board \rightarrow ride \rightarrow exit \rightarrow board \ldots$, when a passenger simply changes his bus a few times. Future work that incorporates an 'elaboration-level' control and more expressive script knowledge representation might be able to alleviate these issues.

5.3.3 Generalizability Aspects

Due to the stochastic nature of the *agenda generator*, the agendas it produces rarely coincide with the ones in the corpus (less than 0.1%). That

means our model can successfully generate new script story lines.

In terms of generalizing to other domains, it is worth noting that events is not the only means of planning a story. Any category of words or symbolic information that could outline a story (conditioned on a specific topic), could take the role of events in our model and allow for the application of our approach. Examples include ingredient usages in a recipe, incidents in a football match, and progresses in a presidential election.

It is well observed that the `InScript` corpus we use contains massive manual annotation effort. However, we note the the event annotations we use is inherently a cluster of utterances that correspond to same script events. Thus it is feasible to substitute the event annotations in our method with predicate-argument structures, which could be acquired by dependency parsing.

6 Conclusion

To incorporate global coherence of story generation on small-scale corpora, we developed a novel, data-driven, hybrid model which exploits a latent intention variable to guide story generation. The model includes a symbolic *agenda generator* that performs text planning and is less demanding on data, and a neural surface realization module that accomplishes surface realization conditioned on the agenda. Our model outperformed various baselines according to the result of a human evaluation experiment which mostly focused on global coherence. The model could be generalized to other domains where words that indicate narration progress are available. Future work will include the exploration of some control over the level of elaboration and developing more expressive script knowledge representation to account for more complicated scripts.

References

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305.

Stephan Busemann and Helmut Horacek. 1998. A flexible shallow approach to text generation. *arXiv preprint cs/9812018*.

Kyunghyun Cho, Bart van Merriënboer Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

François Chollet et al. 2015. Keras. `https://keras.io`.

Angela Fan, Mike Lewis, and Yann Dauphin. 2019. Strategies for structuring story generation. *arXiv preprint arXiv:1902.01109*.

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

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 329–339.

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

Mirella Lapata and Regina Barzilay. 2005. Automatic evaluation of text coherence: Models and representations. In *IJCAI*, volume 5, pages 1085–1090.

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, Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 868–875.

James Richard Meehan. 1976. The metanovel: writing stories by computer. Technical report, YALE UNIV NEW HAVEN CONN DEPT OF COMPUTER SCIENCE.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2015. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. *arXiv preprint arXiv:1509.00838*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Ashutosh Modi, Tatjana Anikina, Simon Ostermann, and Manfred Pinkal. 2017a. Inscript: Narrative texts annotated with script information. *arXiv preprint arXiv:1703.05260*.

Ashutosh Modi, Ivan Titov, Vera Demberg, Asad Sayeed, and Manfred Pinkal. 2017b. Modeling semantic expectation: Using script knowledge for referent prediction. *arXiv preprint arXiv:1702.03121*.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318.

Nanyun Peng, Marjan Ghazvininejad, Jonathan May, and Kevin Knight. 2018. Towards controllable story generation. In *Proceedings of the First Workshop on Storytelling*, pages 43–49.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2018. Data-to-text generation with content selection and planning. *arXiv preprint arXiv:1809.00582*.

Christopher Purdy, Xinyu Wang, Larry He, and Mark Riedl. 2018. Predicting generated story quality with quantitative measures. In *Fourteenth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Florian Pusse, Asad Sayeed, and Vera Demberg. 2016. Lingoturk: managing crowdsourced tasks for psycholinguistics. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 57–61.

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

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

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

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Lilian Wanzare, Alessandra Zarcone, Stefan Thater, and Manfred Pinkal. 2017. Inducing script structure from crowdsourced event descriptions via semi-supervised clustering. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 1–11.

Tsung-Hsien Wen, Milica Gasic, Dongho Kim, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. *arXiv preprint arXiv:1508.01755*.

Denis Yarats and Mike Lewis. 2017. Hierarchical text generation and planning for strategic dialogue. *arXiv preprint arXiv:1712.05846*.

Jian Zhang, Xiaofeng Wu, Andy Way, and Qun Liu. 2016. Fast gated neural domain adaptation: Language model as a case study. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1386–1397.

A Hyper-parameter Tuning

Hyper-parameters are tuned with a two stage random hyper-parameter search. In both stages we test 60 random hyper-parameter combinations; the 5 best-performing hyper-parameter combinations in the first stage decide the ranges from which the hyper-parameters combinations for the second stage were sampled. Table 4 shows the intervals that the hyper-parameters were sampled from in the first stage. Table 5 shows the hyper-parameters that we finally chose. Each training session takes 3 to 4 hours on a single TITAN X.

B More Details on Human Evaluation

Four stories per model variant per script (that is, 200 stories in total) were randomly selected for evaluation. Each task included the assessment of five stories (one from each system); participants were compensated with 1.5GBP per task, which corresponds to a payment of approx. 7GBP per hour. For each of the stories, we collected the judgments of about 10 crowd-sourcing participants (about 400 participations in total). All participants were native English speakers. Submissions that left at least one question unanswered or fall beyond 3 standard deviations are excluded from the statistics. As a result, we received 1221 valid evaluation items in total.

C Automatic Metrics

We attempted a few automatic metrics for evaluating the quality of generated stories proposed in the literature (see Lapata and Barzilay, 2005; Purdy et al., 2018), including `word overlap` (average of word overlap in consecutive sentences), `sentence vector` (average cosine of sentence vectors of consecutive sentences), `coreference rate` (proportion of entities refering to one already mentioned). The results and their correlation with human evaluation are shown in figure 6. Due to their poor correlation with human evaluation results, we decided not to rely on these metrics.

Hyper-parameter	Range	Sampling Criterion
dropout rate	[0.2, 0.8]	uniform*
learning rate	$[10^{-5}, 10^{-3}]$	exponential**
gradient norm threshold	[1.0, 1000.0]	exponential**
batch size	$[2^3, 2^{10}]$	uniform-p2***
context length	[5, 100]	uniform-int****
event embedding size	$[2^6, 2^{10}]$	uniform-p2***
RNN size	$[2^6, 2^{11}]$	uniform-p2***
β: weight on loss term L_a	[1.0, 2.0]	uniform*
γ: weight on category-1 cross-entropy	[1.0, 6.0]	uniform*

*: sampled from a uniform distribution over the range.
**: sampled from a truncated exponential distribution over the range. i.e., we sampled its logarithm from a uniform distribution.
***: sampled from a uniform distribution over the powers of 2 in the range.
****: sampled from a uniform distribution over all integers in the range.

Table 4: The initial ranges and sampling criteria of the random hyper-parameter search.

event embedding size	learning rate	context length	batch size	maximum gradient norm
512	$1.9e{-}5$	46	256	3.17
Dropout	GRU size	β	γ	
0.456	1024	1.01	5.46	

Table 5: The final choices of hyper-parameters. β is the weight applied on the output a and γ is weight applied on the loss of the less frequent category in the binary classification.

System	Word Overlap	Sentence Vector	Coreference Rate
`human author`	0.18(0.020)	0.62(0.019)	0.11(0.015)
`full`	0.27(0.030)[2]	0.89(0.029)[1]	0.00061(0.00070)
`random event order`	0.26(0.015)[2]	0.90(0.023)[1]	0.002(0.002)
`GRU+Topic`	0.35(0.11)	0.74 (0.078)	0.18(0.066)
`GRU`	0.26(0.018)	0.69(0.0080)	0.23(0.038)
corelation with human evaluation	-0.59	-0.05	-0.55

[1],[2]: differences between pairs are not statistically significant according to pair T-tests.

Table 6: Results from automatic evaluation, and their correlation with overall human evaluation results. Encoding some information about the text though they may, these scores are hardly informative about global coherence.