Guiding Neural Story Generation with Reader Models

Xiangyu Peng, Kaige Xie, Amal Alabdulkarim, Harshith Kayam, Samihan Dani, Mark O. Riedl
Georgia Institute of Technology
{xpeng62,kaigexie,amal,hkayam3,sdani30,riedl}@gatech.edu

Abstract

Automated storytelling has long captured the attention of researchers for the ubiquity of narratives in everyday life. However, it is challenging to maintain coherence and stay on-topic toward a specific ending when generating narratives with neural language models. In this paper, we introduce Story generation with Reader Models (StoRM)[1], a framework in which a *reader model* is used to reason about the story should progress. A reader model infers what a human reader believes about the concepts, entities, and relations about the fictional story world. We show how an explicit reader model represented as a knowledge graph affords story coherence and provides controllability in the form of achieving a given story world state goal. Experiments show that our model produces significantly more coherent and on-topic stories, outperforming baselines in dimensions including plot plausibility and staying on topic.

1 Introduction

Automated Story Generation is the challenge of designing an artificial intelligence system that can generate a natural language text that is perceived by readers as a story. Early work on story generation used symbolic planning (Meehan, 1976; Lebowitz, 1987; Cavazza et al., 2003; Porteous and Cavazza, 2009; Riedl and Young, 2010; Ware and Young, 2010; Ware and Siler, 2021). These systems would be provided with a description of the initial world state—usually a list of predicates—and a goal—a description of what predicates should be true to be successful. These approaches had two benefits. First, the plots tended to be coherent because of logical constraints on the actions. Second, the plots were guaranteed to end in a state in which the goal held. However, these systems require substantial knowledge engineering of logical constraints, limiting their generality, and don't always generate plot or stories in natural language.

[1]Code: https://github.com/xiangyu-peng/Reader_Model

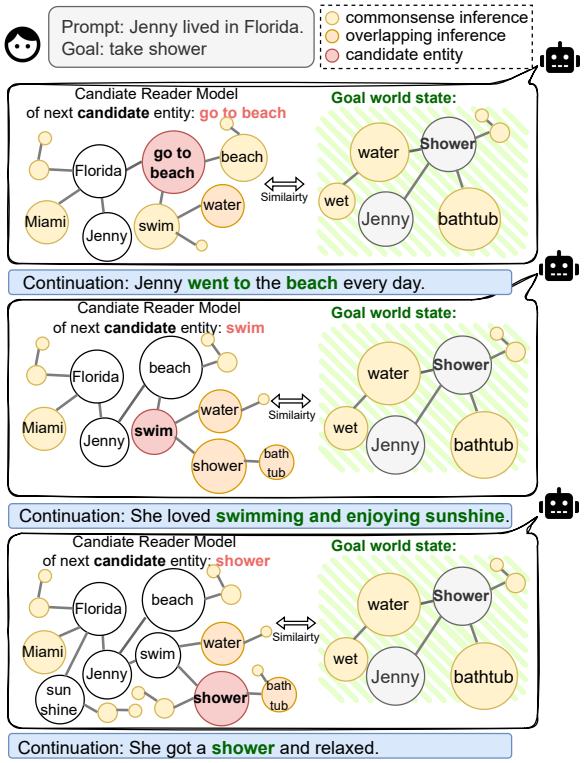

Figure 1: The overview of StoRM system. Our goal is to generate a story on a prompt for reaching the given goal. 1. The system builds a goal world state by converting natural language text into knowledge graph and then expands it with commonsense inference . 2. StoRM builds a prompt story world and then infers a set of concepts on each entity in prompt story world. 3. For every concept , StoRM obtains a candidate reader model by inferring commonsense relations on it. 4. StoRM chooses the *topk* concepts which maximize its similarity with goal world state. 5. A language model generates continuation options on the *topk* concepts, which are added to the story.

Recently, neural language modeling approaches (Roemmele, 2016; Khalifa et al., 2017; Martin et al., 2018; Clark et al., 2018; Yao et al., 2019; Rashkin et al., 2020; Fan et al., 2019; Ammanabrolu et al., 2021a) have been applied to story

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 7116 - 7140
December 7-11, 2022

	ROCStories	Writing Prompts	Fairytale
StoRM	**Charles had always wanted to have a college degree.** He was admitted to a college. Finally, he graduated from college. He was glad he finally **got his diploma**.	**Jennifer wander in hallway.** She thought she had found something on the ground. When she looked at the ground, there was nothing. She then kept walking in the dark. She cannot see any light. Then, **she fell on ground**. She was in a lot of pain. When she got back up, she fell again. This time, she got back up and just walked away. "Hello?" she asked. **She walked toward kitchen.**	**Hero consumed them.** Then, he got very sick and died. He buried himself. Then, he reborned. **He showed up.** He made a speech to the people. He received applause from the audience. After, he thanked the audience for their support. When he finally came back, everyone cheered. **Hero invoked command to his magic sword.** Immediately, he became more powerful than ever. Now, he was the boss of the world. He began to be in charge of the world. He would be a king. **He had again assumed his place as King of Golden Mountain.**
Baselines (C2PO or CP)	**[C2PO] Charles had always wanted to have a college degree.** Charles begins get a job. Charles tries to work hard. Charles wants to get a degree. **Charles was proud that he had a college degree.**	**[CP] Jennifer wander in hallway.**"Hey," said the woman in front of her. "Is this the elevator? I think it is." "I don't think of that. It looks like its the elevator." "Jennifer asked you 're the only thing keeping the girl alive", The girl said. **Jennifer fell on ground.** She was on her way home from work when she saw a strange figure in the kitchen. She turned. "Hey." Jennifer said to an odd looking man. All he saw was a girl that looked like it that looked back at him. **Jennifer walked toward kitchen.**	**[C2PO] Hero consume them.** Hero starts to eat something. Hero wants drink water. Hero begins to talk to someone. Hero begins to have a reason. **Hero revealed himself.** Hero starts to tell everyone about it. Hero wants to practice. Hero tries to know magic. **Hero invoked command to his magic sword.** Hero wants to prove his skills. Hero wants to show off. Hero wants to be in charge. Hero wants to be king. **He had again assumed his place as King of Golden Mountain.**

Table 1: Story examples generated by StoRM, C2PO, and Goldfarb-Tarrant et al. (2020) (CP). Stories generated on ROCStories and Fairytale stories are by C2PO and the story on Writing Prompts is generated by Goldfarb-Tarrant et al. (2020). *Bolded* text are prompts and goal texts. Underlined text indicates early stopping when the generated story hits the goal. More details about generation can be found in Appendix D.

generation because they circumvent the need for manual knowledge engineering and tend to produce relatively fluent, varied, and naturalistic language. Language models are, however, not goal-directed. That is, one cannot natively provide both a context prompt and a goal to be achieved after an arbitrary number of continuations. Further, language models struggle with maintaining story coherence—the logical progression of events—and may also become repetitive. Large, pre-trained language models improve fluency and generalization but do not provide goal-directedness and stories generated can still be perceived as lacking in coherence in the sense that they meander without direction.

We consider the challenge of coherent and controllable text generation for neural language model based story generation. We hypothesize that neural language models, while powerful text-completion systems, are not natively well-suited for coherent story generation because a neural network trained with a cross-entropy loss function is unlikely to model the unfolding context of a story the same way as a human reader. Studies of human reader comprehension (Zwaan and Radvansky, 1998) show that readers comprehend stories by tracking the relations between entities and events in ways that can be expressed as a graph. The perceived coherence of a story is a function of the connectedness of this graph (Graesser et al., 1994). Ensuring the causality between sentences can improve the coherence of stories (Peng et al., 2021).

Inspired by cognitive science, we aim to augment neural language models with a **reader model** in which a story generation system infers a graph of concepts, entities, and relations that a reader is likely to believe about the story world as they read an incrementally generated story. The reader model enables the story generation algorithm to explicitly reason about the entities and relations and generate story continuations that use those entities to move the story forward; a reader can track how entities and relations change over time and thus perceive stories as more coherent. We use large language models to produce the continuation text of the story generation. However instead of only providing the previous story as context, our algorithm also selects one or more entities from the world model and uses template filling to generate candidate continuations.

The reader model also provides a means for directing the generation process. In addition to a starting context prompt, we require a goal to be given. The goal is natural language text which is then converted into a knowledge graph (See Fig. 1).

The goal knowledge graph provides a rough outline of the entities and relations that need to be present in the story but without providing particulars about everything that must be in the story or the ordering in which they must occur.

Our contributions are as twofold: (1) we propose an automated story generation model with Reader Models (**StoRM**) which maintain coherence and controllability of generated stories at the same time; and (2) we conduct a thorough experimental study against strong baselines which shows that StoRM produces significantly more coherent and goal-directed story.

2 Related Work and Background

We situate our paper in the literature of neural networks—recurrent and transformer-based—to produce stories (Roemmele, 2016; Khalifa et al., 2017; Martin et al., 2018; Clark et al., 2018; Fan et al., 2018). There are a few works that are highly related to our proposed framework, in terms of the following two dimensions: the generation controllability and the usage of commonsense knowledge. The controllability in story generation focuses on how to enable the generation process to adhere to the user's inputs. Goldfarb-Tarrant et al. (2020) conducts generation in two steps: planning a story plots based on a prompt, then generating a story by filling the mask tokens in plots with BART(Lewis et al., 2020). Plot Machines (Rashkin et al., 2020) accepts as an input an un-ordered outline of concepts and conditions a language model.

Commonsense knowledge plays an important role in story generation. The most popular way of utilizing it is to train neural language models (e.g. GPT-2 (Radford et al., 2019)) on commonsense knowledge bases such as ConceptNet (Speer and Havasi, 2013) and ATOMIC (Sap et al., 2019; Hwang et al., 2021) which contains detailed information regarding well-known facts or causal relationships. Thus the resulting language model, named COMET (Bosselut et al., 2019; Hwang et al., 2021), becomes capable of inferring new commonsense knowledge on novel phrases. Given a sentence, COMET infers commonsense attributes about the characters in three categories: (1) social interactions (i.e. `xNeed` indicates what the character of this sentence needed), (2) physical entities, and (3) effect of events inferred from the sentence. Ammanabrolu et al. (2021b) proposes Causal, Commonsense Plot Ordering (C2PO) framework which takes advantage of COMET to infer predecessor and successor events and then bi-directionally search from pre-specified start event to end event, however, C2PO generates plots made up of highly constrained, templated text; Peng et al. (2021) leverages COMET to infer the character intentions and effects of actions so as to guide the generation process, but they did not consider controllability. There are also other approaches that directly incorporate commonsense knowledge graphs into the encoding process (Mihaylov and Frank, 2018; Guan et al., 2019). Compared with them, the novelty of our paper is to improve coherence and controllability at the same time with the help of external commonsense knowledge graph.

3 Story Generation with Reader Models

In this section, we introduce a framework—*Story generation with Reader Models* (StoRM)—for generating stories with models of what the reader will believe about the fictional story world. We hypothesize that the incorporation of a *reader model* into the story generation process will increase story coherence. We define *story coherence* as the extent to which readers can identify connections between different events and entities in a story. In this work, the reader model is represented as a *knowledge graph*, a set of triples of the form $\langle subject, relation, object \rangle$. By making the beliefs about what the reader likely knows explicit, we provide mechanisms for selecting which entities to include in the continuation of the story. Because the StoRM framework maintains a knowledge graph that approximates the reader's beliefs about the story world, we are able to compare the reader model to a desired goal world state, also described as a knowledge graph. The StoRM framework is thus *controllable*.

Our framework starts with a prompt and a set of goals (See Fig. 2), which are natural language texts. The prompt and goals are transformed into knowledge graphs by extracting entities (§3.1), referred as reader model and goal world state (see yellow bubbles in Fig. 2). These two knowledge graphs are expanded using commonsense techniques (§3.2; See red bubbles in Fig. 2). For every inferred entity in the reader model, we further expand the reader model with its commonsense inference to obtain a candidate knowledge graph, which is scored based on how similar it is relative to the goal world state (§3.3). Entities of which the candidate KG shares

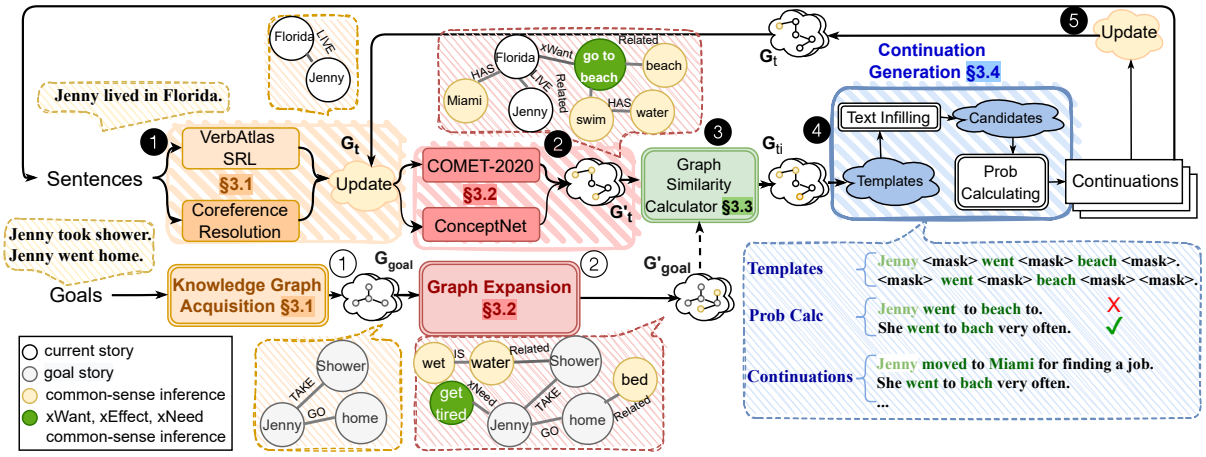

Figure 2: The overall procedure of StoRM. ❶ ① Prompts and goals are transformed into knowledge graph (§3.1). ❷ ② They are expanded with inferences (§3.2) and obtain a set of candidate knowledge graphs $\mathbf{G}'_t$. ❸ $\mathbf{G}'_t$ are compared to and *topk* candidate knowledge graphs are kept (§3.3). ❹ Generate story continuations on *topk* candidate knowledge graphs (§3.4). ❺ Story continuations are appended to story history and also update candidate knowledge graphs with details in continuations.

the *topk* similarity with the goal world state are selected and the generation technique uses templates to generate possible story continuations (§3.4; See blue bubbles in Fig. 2). By targeting different entities and using template infilling, we reduce neural network hallucination of new entities and create a diverse set of story continuations. The selected continuation starts the next iteration of the generation process. Story examples can be found in Table 1.

3.1 Knowledge Graph Acquisition

With the automatic generation of the story, some important information could be forgotten. The knowledge graph is an explicit and *persistent* memory of entities mentioned or inferred from the story text generated so far. Knowledge Graphs represent information in the form of triples, consisting of a subject entity, relation and object entity. For example, "*Jenny lived in Florida*" is represented as $\langle jenny, live, florida \rangle$. The entities represent the nodes of the graph and their relations act as edges.

To acquire the knowledge graph, we firstly trained a Semantic Role Labeling (SRL) model (Gildea and Jurafsky, 2002) on VerbAtlas (Di Fabio et al., 2019)—a hand-crafted lexical-semantic resource whose goal is to bring together all verbal synsets from WordNet (Fellbaum, 1998) into semantically-coherent frames. This SRL model provides the automatic identification and labeling of argument structures of stories. Further detail of training can be found in Appendix B.1.

StoRM then converts the output of VerbAtlas SRL model into knowledge graph triples. Entities represent the theme and attribute and VerbAtlas frames act as edges. An example is shown in yellow bubbles of Fig. 2. Multiple character names, object names and pronouns make the knowledge graph representation hard to interpret. Hence, we adopt a end-to-end Coreference Resolution model (Lee et al., 2017) to find all expressions that refer to the same entity in a story to minimize the entities.

StoRM starts with two knowledge graphs. The first, $\mathbf{G}_1$, is the converted prompt (first sentence). The second $\mathbf{G}_{\text{goal}}$ is the converted goal. With the generation of the continuation candidates, we will update the knowledge graph $\mathbf{G}_t$ with new continuations to get new knowledge graph $\mathbf{G}_{t+1}$, where t is the index of the sentence in the story. Because we obtain the *topk* continuation knowledge graphs, hence $\mathbf{G}_t = \{\mathbf{G}_{t1}, ..., \mathbf{G}_{tK}\}$.

3.2 Graph Expansion

Human readers use commonsense knowledge to infer the presence of entities and concepts not explicitly mentioned in story text. For example, "Florida" has "beaches" and "eating dinner" implies "dishes". In accordance, we use common-sense inferences of entities to expand the knowledge graph to provide entities for characters to interact with and thus drive the story forward. Because the presence of entities and concepts are inferred from prior events, the reader should be able to track the connections between entities and events, thus supporting perceived story coherence.

We consider two types of nodes in knowledge graph—physical entities (i.e. "beach") and social events (i.e. "go to beach"). We use ConceptNet5 (Speer and Havasi, 2013)—a multilingual knowledge base, representing words and phrases that people use and the common-sense relations between them—to infer each physical entity. We use COMET$^{20}_{20}$ (Hwang et al., 2021)—a transformer-based generative model trained on the ATOMIC$^{20}_{20}$ commonsense dataset (Hwang et al., 2021)—to infer relations on events about commonsense relations of social interaction. More details about inference types can be found in Appendix B.2.

Firstly, we expand the physical entities in current knowledge graph $\mathbf{G}_{ti}$ to obtain a *physical entity candidate set* $E_{ti,\text{entity}}$ with ConceptNet (i.e. expand "Florida" with "Miami"; yellow nodes in Fig.2). We then expand each entity $e^{k_1}_{ti,\text{entity}} \in E_{ti,\text{entity}}$ with ConceptNet until we reach a depth of m_1 to obtain $\widetilde{E^{k_1}_{ti,\text{entity}}}$, where $k_1 \in [1, n_1]$ and n_1 is the size of $E_{ti,\text{entity}}$. Secondly, we generate social interactions of current story history by COMET$^{20}_{20}$ to obtain an *social event candidate* set $E_{ti,event}$ (i.e. we infer "Jenny lived in Florida" to "go to beach"; green nodes in Fig. 2) . We expand each event $e^{k_2}_{ti,event} \in E_{ti,event}$ with COMET$^{20}_{20}$ and ConceptNet until we reach a depth of m_1 to obtain $\widetilde{E^{k_2}_{ti,event}}$, where $k_2 \in [1, n_2]$ and n_2 is the beam size of the output of COMET$^{20}_{20}$. Hence, for $\mathbf{G}_{ti}$, we totally obtain n entity candidates $e^k_{ti} \in \{E_{ti,\text{entity}} \cup E_{ti,event}\}$, where $k \in [1, n]$ and $n = n_1 + n_2$ and its corresponding knowledge graph candidates $\mathbf{G}'_{ti} = \{\mathbf{G}'^1_{ti}, ..., \mathbf{G}'^n_{ti}\}$, where $\mathbf{G}'^k_{ti} = \mathbf{G}_{ti} \cup \widetilde{E^k_{ti}}$ ($\widetilde{E^k_{ti}} = \widetilde{E^k_{ti,\text{entity}}}$ for $e^k_{ti} \in E_{ti,\text{entity}}$; $\widetilde{E^k_{ti}} = \widetilde{E^k_{ti,\text{event}}}$ for $e^k_{ti} \in E_{ti,\text{event}}$). Totally, we obtain $\mathbf{G}'_t = \{\mathbf{G}'_{t1}, ..., \mathbf{G}'_{tK}\}$.

We also expand the depth of the graph for goals by m_2 via similar means: we firstly expand all the physical entities in $\mathbf{G}_{\text{goal}}$ by m_2 depth with ConceptNet to obtain inference set $\widetilde{E_{\text{goal,enitity}}}$. Then we expand social interactions of the goal text with COMET$^{20}_{20}$ and ConceptNet by m_2 depth to obtain event candidates $\widetilde{E_{\text{goal,event}}}$. Finally, we obtain the updated knowledge graph $\mathbf{G}'_{goal} = \mathbf{G}_{\text{goal}} \cup \widetilde{E_{\text{goal,enitity}}} \cup \widetilde{E_{\text{goal,event}}}$. Further details can be found in Appendix A.

3.3 Graph Similarity

We achieve controllability of generated continuations by calculating the *graph similarity* between the candidate knowledge graph $\mathbf{G}'^k_{ti}$ and the goal knowledge graph $\mathbf{G}'_{\text{goal}}$. The candidate knowledge graph $\mathbf{G}^k_{ti}$ is updating $\mathbf{G}_{ti}$ with candidate entity e^k_{ti}.

We calculate the knowledge graph similarity score:

$$R(\mathbf{G}^k_{ti}) = \alpha \times r_1(\mathbf{G}^k_{ti}, \mathbf{G}_{\text{goal}}) + (1-\alpha) \times r_2(\widetilde{E^k_{ti}}, \mathbf{G}'_{\text{goal}})) \quad (1)$$

where r_1 is story entity overlapping score and r_2 is inference overlapping score. α is a hyperparameter to control the inference's contribution on calculating overlapping rate. $\widetilde{E^k_{ti}}$ is inference set of candiate entity e^k_{ti}.

Story entity overlapping score (r_1) calculates the overlapping rate between the candidate knowledge graph $\mathbf{G}^k_{ti}$ and the full knowledge graph $\mathbf{G}_{\text{goal}}$ without considering inference nodes (white and gray nodes in Fig. 2). We define a match as same entities (nodes) between two knowledge graph. Then calculate the story entity overlapping rate by

$$r_1(\mathbf{G}^k_{ti}, \mathbf{G}_{\text{goal}}) = \frac{\sum_j \sum_l \mathbb{I}(e_{j,\mathbf{G}^k_{ti}} = e_{l,\mathbf{G}_{\text{goal}}})}{\text{size of } \mathbf{G}_{\text{goal}}} \quad (2)$$

where $\mathbb{I}(e_{j,\mathbf{G}^k_{ti}} = e_{l,\mathbf{G}_{\text{goal}}}) = 1$ when there is a match between entity $e_{j,\mathbf{G}^k_{ti}} \in \mathbf{G}^k_{ti}$ and entity $e_{l,\mathbf{G}_{\text{goal}}} \in \mathbf{G}_{\text{goal}}$, otherwise 0.

We then calculate the overlapping rate between $\widetilde{E^k_{ti}}$ and goal knowledge graph with inferences, $\mathbf{G}'_{\text{goal}}$, as *inference overlapping score* r_2,

$$r_2(\widetilde{E^k_{ti}}, \mathbf{G}'_{\text{goal}}) = \frac{\sum_j \sum_l \mathbb{I}(\hat{e}_{j,\widetilde{E^k_{ti}}} = e_{l,\mathbf{G}'_{\text{goal}}})}{\text{size of } \mathbf{G}'_{\text{goal}}} \quad (3)$$

where $\hat{e}_{j,E^{si}_t} \in \widetilde{E^k_{ti}}$ and $e_{l,\mathbf{G}'_{\text{goal}}} \in \mathbf{G}'_{\text{goal}}$.

We also consider constructing an inference link between current story and the goal. When we generate $\widetilde{E^k_{ti,\text{event}}}$ and $\widetilde{E_{\text{goal,event}}}$ in Section 3.2, we obtain $\sum_{j=1}^{m_1} {n_2}^j$ and $\sum_{j=1}^{m_2} {n_2}^j$ inference links, where n_2 is the beam size of the output of COMET$^{20}_{20}$. For example, when generating $\widetilde{E_{\text{goal,event}}}$, the `xNeed` of the goal "enjoy sunshine" is "go to beach". Then we develop a link from goal—"go to beach → enjoy sunshine". When generating $\widetilde{E^k_{ti,\text{event}}}$, the `xWant` of the prompt "live in

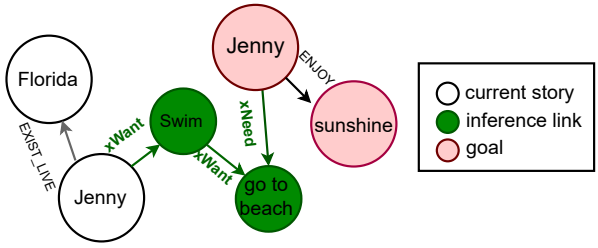

Figure 3: A demonstration of finding the inference link between current story and the goal.

Florida" is "swim" and the `xWant` of "swim" is "go to beach". Then we successfully develop a link between the prompt and the goal—"live in Florida $\rightarrow$ swim $\rightarrow$ go to beach". An example is shown in Fig. 3.

Hence, we calculate the semantic similarity between all events in $\widetilde{E^k_{ti,\text{event}}}$ and $\widetilde{E_{\text{goal,event}}}$ by Sentence-BERT (Reimers and Gurevych, 2019). When the max semantic similarity score is over threshold[2], we set $R(\mathbf{G}^k_{ti}) = 1$ and generate stories based on this link. We obtain the *topk* continuation knowledge graphs with the *topk* highest graph similarity scores. They will be used to produce continuations further. We always keep a total of K knowledge graphs (reader models) when generating stories for each index of the sentence in the story. Thus the full generation process is implemented as a form of beam search.

3.4 Continuation Candidate Generation

Given *topk* continuation knowledge graph $\mathbf{G}^k_{ti}$ and its corresponding event e^k_{ti}, we generate story continuations. We consider the conditional sentence generation as a infilling task (Taylor, 1953).

Templates. A set of templates T^k_{ti} are generated on each event e^k_{ti}. For example, one of the templates generated on `swim` are `[subject] <mask> <mask> swim <mask>`. The `[subject]` of the sentence is (1) the same subject with the previous sentence, (2) no fixed (`<mask>`), or (3) any characters in previous story history (See Appendix B.3).

Text Infilling. We fine-tune RoBERTa (Liu et al., 2019) on story datasets (§4). Details are shown in Appendix B.4. All the templates T^k_{ti} are filled by fine-tuned RoBERTa and we obtain a number of continuation candidates $S^k_{(t+1)i} = \{s^k_{(t+1)i,1}, ...s^k_{(t+1)i,m}\}$ for each event e^k_{ti} where m is the number of templates of each entity.

[2]we use 80% in this paper.

Filtering. We fine-tune GPT-2 (Radford et al., 2019) on different datasets(See §4) and filter the continuation candidates by calculating their conditional probability $\mathbb{P}_s$ with it: $\mathbb{P}_s = \prod_{j=1}^{n} \mathbb{P}(\mathrm{X}_j|\mathrm{X}_1, ..., \mathrm{X}_{j-1})$,where n is the length of the sentence s and X_j is the jth token in sentence s. We only keep one sentence $s^k_{(t+1)i} \in S^k_{(t+1)i}$ with the highest probability for each event e^k_{ti} and append $s^k_{(t+1)i}$ to the ith story, where $i \in [1, K]$.

4 Experiments

After proving that the knowledge graph acquisition technique is able to capture the information that natural language story conveys in Appendix C. We evaluate our system with two experiments, comparing StoRM to two strong neural language model story generators on the the dimensions of coherence and controllability.

Datasets. We conduct the experiments on three datasets:

- ROCStories(Mostafazadeh et al., 2016): contains 98,159 five-sentence stories involving commonsense scenarios. The first and last sentence of one story are used as the prompt and the goal, respectively.
- Writing Prompts (WP) (Fan et al., 2018): a large collection of user-generated stories along with their associated prompts from Reddit. Average story length is 59.35 sentences, which provides longer and complicated stories than ROCStories. We extract high-level plots following Ammanabrolu et al.(2020a), which are used as prompts and goals to generate stories.
- Fairy tale stories (FT): following (Ammanabrolu et al., 2020b), we scraped 695 stories in fairy tales genre from Wikipedia, which provides a new genre of stories. Average story length is 24.80 sentences. Same with Writing Prompts, extracted high-level plots are used as prompts and goals to generate stories.

Baselines. We evaluate our model against two strong baselines:[3]

- C2PO(Ammanabrolu et al., 2020a): uses COMET(Bosselut et al., 2019) to generate successor and predecessor events, performing a bidirectional search from a given start event and a

[3]Two potential baselines were considered but not pursued. Tambwekar et al. (2019) is goal-driven but does not produce natural language without manual intervention. The system by Rashkin et al. (2020) accepts unordered outline terms but the results of the original paper could not be reproduced at the time of writing.

given end event. It uses a subset[4] of social interaction relations generated by COMET to build a commonsense link between the prompt and the goal, without constraining language model. For fair comparison, following Ammanabrolu et al., StoRM and C2PO generate a story piece-by-piece given a set of goals. Details about this baseline can be found in Appendix B.5.

- Goldfarb-Tarrant et al. (2020): trains BART(Lewis et al., 2020) to generate plots on the given prompt and then transform it into a story with an ensemble of rescoring model on Writing Prompts dataset. For fair comparison, we set StoRM and Goldfarb-Tarrant et al. (2020) up for generating a story piece-by-piece given a set of goals. We firstly follow Ammanabrolu et al. (2020a), to extract high-level plots from Writing Prompts. We then train the model on the Writing Prompts using these extracted plots as the prompts, and sections in between each of these extracted plot points as the story. More details about training can be found in Appendix B.6.

4.1 Story Coherence Evaluation

We firstly seek to understand whether StoRM improves the coherence and quality of the generated story. To generate the stories, we randomly selected 15 stories from each dataset and condition StoRM and baselines with same prompts and goals. More details can be found in Appendix D.

Human Evaluation. We first evaluate coherence using human participant evaluation, asking a set of questions that includes dimensions such as logical coherence, enjoyability and fluency. Variations of these questions have been used to evaluate other story generation systems (cf. (Purdy et al., 2018; Tambwekar et al., 2019; Ammanabrolu et al., 2020c, 2021a; Castricato et al., 2021; Peng et al., 2021)). We focus on dimensions involving overall perceptions of narrative coherence:

- The story's sentences MAKE MORE SENSE given sentences before and after them: evaluates local causality and commonsense reasoning.
- The story is more ENJOYABLE: indicates story value.
- The story uses more FLUENT language: indicates story readability.

Each human participant reads a randomly selected subset of story pairs, comprised of one story from StoRM and one from baselines. For the above three questions, participants answered which story best met the criteria. Details about human study can be found in Appendix E.2.

Table 2.A (left) shows the percentage of times stories from each system are preferred for each metric. StoRM improves the perception of narrative coherence of generated narratives, and also produce more enjoyable and fluent stories with the help of common-sense relations. With the help of transformer-based language model, Goldfarb-Tarrant et al. (2020) achieves comparable enjoyability with StoRM but fail to guarantee local causality in the story. C2PO only applies a limited number of ATOMIC relations to conduct a bi-directional search between the prompt and the goal, which cannot guarantee coherence between all the sentences in the story. At the same time, the generated stories by C2PO are made up of highly constrained, templated text, which also decrease the fluency and enjoyability of the story.

Automatic Metrics. We also evaluate diversity of generated stories by measuring Self-BLEU scores (Zhu et al., 2018). For each generated story, we take one sentence as the hypothesis and the others as references and calculate the BLEU score, repeating for every sentence in the story. The averaged BLEU score of its generated stories is defined as the self-BLEU score of the model. A lower self-BLEU score indicates more diversity of the stories. Table 2.A (right) shows that StoRM significantly outperforms C2PO on the dimension of "diversity", because StoRM adopts transformer-based language model to generate stories, however, C2PO uses templated and limited range of COMET to generate stories. Compared with Goldfarb-Tarrant et al. (2020), which also applies transformer-based language model, StoRM has comparable results on the diversity of generated stories.

4.2 Controllability Evaluation

We assess whether StoRM is able to achieve the given goal, as measured by human evaluation and automated metrics. Generated stories evaluated in Section 4.1 are reused for evaluating controllability.

Human Evaluation. Human participant firstly read a prompt and a goal in natural language. They then read a randomly selected generated story pairs—one from StoRM and one from our baselines. They then answered which one better met the criteria (See Appendix E.3):

[4] C2PO uses `xNeed` and `xWant` relations.

Models	Data set	Logical Sense			Enjoyable			Fluency			Self-BLEU-2↓		Self-BLEU-3↓	
		Win%	Lose%	Tie%	Win%	Lose%	Tie%	Win%	Lose%	Tie%	StoRM	Baseline	StoRM	Baseline
StoRM vs CP	WP	**72.0****	18.7	9.3	**52.0**	40.0	8.0	**74.7****	18.7	6.7	.137	**.103**	.098	**.070**
StoRM vs C2PO	ROC	**72.9****	17.1	10.0	**71.4****	21.4	7.1	**62.9****	24.3	12.9	**.045****	.261	**.035****	.169
	FT	**58.7****	22.7	18.7	**60.0****	22.7	17.3	**53.3****	21.3	25.3	**.095****	.249	**.072****	.155

Table 2.A Evaluation results on coherence and diversity.

Models	Data set	Goal			Quality			BLEU-2↑		BLEU-3↑		ROUGE-L↑		S-M↑	
		Win%	Lose%	Tie%	Win%	Lose%	Tie%	StoRM	Baseline	StoRM	Baseline	StoRM	Baseline	StoRM	Baseline
StoRM vs CP	WP	**57.1***	31.4	11.4	**54.3**	38.6	7.1	**.204***	.099	**.169***	.070	**.315***	.244	**.065**	.046
StoRM vs C2PO	ROC	**73.3****	16.0	10.7	**74.7****	20.0	5.3	.334	**.361**	.290	**.329**	**.428**	.426	**.121**	.119
	FT	**56.0***	33.3	10.7	**61.3****	26.7	12.0	**.110**	.089	**.079**	.065	**.198**	.182	.044	**.058**

Table 2.B Evaluation results on controllability and coverage with respect to gold stories.

Table 2: Evaluation results, showing the percentage of participants who preferred the first system, second system, or thought the systems were equal. CP indicates (Goldfarb-Tarrant et al., 2020). Each system is conditioned on the same test-set prompts and same goal. * indicates results are significant at $p < 0.05$ confidence level; ** at $p < 0.01$ using a Wilcoxan sign test on win-lose pairs. See results about majority votes and agreement in Appendix F.

- Which story better FOLLOWS A SINGLE TOPIC TO ACHIEVE GOAL: evaluates perceptions of global coherence for the entire story.
- Which story has higher QUALITY to achieve the goal: measures overall perceived story quality.

Table 2.B (left) shows that, given the same goal and prompt, StoRM performs significantly better than C2PO on producing more goal-directed and higher-quality stories with fair agreement (See Appendix F). C2PO restricts its bi-directional search from a given start event and an end event in a subset of ATOMIC relations generated by COMET. Hence, it often fails to find a coherent commonsense link to develop a story, which makes human participants hard to understand (see Table 1).

StoRM also significantly outperforms Goldfarb-Tarrant et al. (2020) on the dimension of "Goal", with fair agreement. Compared to Goldfarb-Tarrant et al. (2020)—which applies transformer-based language model to learn how to achieve a goal, StoRM successfully produce a much more goal-directed story with the help of external commonsense knowledge. On the dimension of "quality", StoRM is preferred but the result is not statistically significant when ties are considered, which indicates StoRM improves controllability while retaining high quality of stories.

Automatic Metrics. We also evaluate controllability by measuring the the following three metrics with respect to the gold stories.

- *Sentence mover's similarity (S-M)* (Clark et al., 2019): Evaluate stories in a continuous space using word and sentence embeddings. Larger sentence mover's similarity indicates higher similarity between generation and gold story.
- *BLEU scores*(Papineni et al., 2002): 2-gram, 3-gram BLEU scores are reported.
- ROUGE-L (Lin, 2004) scores: Higher score indicates better coverage.

Right side of Table 2.B shows that StoRM significantly outperforms Goldfarb-Tarrant et al. (2020) in the coverage with respect to gold stories in Writing Prompts dataset. It justifies our human evaluation results that StoRM generates much more goal-directed stories. Compared with C2PO on ROCStories and fairy tale stories, StoRM achieved the comparable results. The story generation is guided by goal events, so it is likely to produce very different story from gold stories, but with same goal (see Table 1).

5 Conclusions

Neural language models are widely used to produce text, including stories. However, they struggle with maintaining *story coherence*—the logical progression of events—and goal-directedness. Our framework—*Story Generation with Reader Models* (StoRM)— augments neural language models with a reader model. This reader model—an explicit knowledge graph—approximates the reader's beliefs about the story world. StoRM increases the story coherence by producing continuations that directly reference entities in this reader model. Goal-directedness is achieved by choosing continuations that add desired entities to the reader's inferred set of beliefs. A thorough experimental study shows

that StoRM produces significantly more coherent and goal-directed stories than two strong baselines on three datasets.

6 Broader Impact

Our system faces the same potential pitfalls as other contemporary language learning systems. It is prone to echoing the biases present in the dataset (Sheng et al., 2019) and generate non-normative text (i.e. in violation of social norms). No existing automated storytelling systems is able to entirely eliminate these biases, though stories can be used to teach language models to reduce non-normative continuations (Peng et al., 2020). Fictional stories that are presented to readers as non-fictional can be used to influence (Green and Brock, 2000) or misinform. Future work may enable real-world facts to be injected into the knowledge graph of a similar system for the purposes of journalism or misinformation. However, because our graph expansion method relies on ConceptNet5 (Speer and Havasi, 2013) and COMET$^{20}_{20}$ (Hwang et al., 2021) for inference, our system is prone to process and produce simple stories.

The ability to produce coherent and goal-directed stories has downstream applications beyond automated story-telling. In particular, this is the first work that increases generative coherence and control by reasoning about the changes to the knowledge graph.

7 Limitations

Restricted by our graph expansion models—COMET$^{20}_{20}$ (Hwang et al., 2021) and ConceptNet5 (Speer and Havasi, 2013), our system works mostly for narratives with event-centric commonsense knowledge. It is also prone to produce and process simple stories.

Additionally, limited by generating continuations by text infilling methods with low GPU resources, our system is best suited for short sentences of 15 words or less. We hope that, with the increase of language model size, graph expansion depth, and continuation template length, our system will benefit from them.

The performance of StoRM is highly related to the inference abilities of COMET$^{20}_{20}$. Hence, the types of errors that COMET is prone to are also the types of errors that our system is prone to. More detailed analysis of commonsense inference errors can be found in Hwang et al. (2021). As more advanced commonsense inference models develop, StoRM-like approaches will benefit from the improved state-of-the-art. StoRM can easily switch to new generative language models or new commonsense inference model.

References

Prithviraj Ammanabrolu, Wesley Cheung, William Broniec, and Mark O. Riedl. 2020a. Automated storytelling via causal, commonsense plot ordering. *CoRR*, abs/2009.00829.

Prithviraj Ammanabrolu, Wesley Cheung, William Broniec, and Mark O Riedl. 2021a. Automated storytelling via causal, commonsense plot ordering. In *Proceedings of AAAI*, volume 35.

Prithviraj Ammanabrolu, Wesley Cheung, William Broniec, and Mark O Riedl. 2021b. Automated storytelling via causal, commonsense plot ordering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 5859–5867.

Prithviraj Ammanabrolu, Wesley Cheung, Dan Tu, William Broniec, and Mark Riedl. 2020b. Bringing stories alive: Generating interactive fiction worlds. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 16, pages 3–9.

Prithviraj Ammanabrolu, Ethan Tien, Wesley Cheung, Zhaochen Luo, William Ma, Lara J Martin, and Mark O Riedl. 2020c. Story realization: Expanding plot events into sentences. In *Proceedings of AAAI*, volume 34.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. Comet: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779.

Louis Castricato, Spencer Frazier, Jonathan Balloch, and Mark O. Riedl. 2021. Tell me a story like i'm five: Story generation via question answering. In *Proceedings of the 3rd Workshop on Narrative Understanding*.

Marc Cavazza, Olivier Martin, Fred Charles, Steven J Mead, and Xavier Marichal. 2003. Interacting with virtual agents in mixed reality interactive storytelling. In *International Workshop on Intelligent Virtual Agents*, pages 231–235. Springer.

Elizabeth Clark, Asli Celikyilmaz, and Noah A Smith. 2019. Sentence mover's similarity: Automatic evaluation for multi-sentence texts. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2748–2760.

Elizabeth Clark, Yangfeng Ji, and Noah A. Smith. 2018. Neural text generation in stories using entity representations as context. In *Proceedings of NAACL-HTL*, pages 2250–2260.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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)*, pages 4171–4186.

Andrea Di Fabio, Simone Conia, and Roberto Navigli. 2019. Verbatlas: a novel large-scale verbal semantic resource and its application to semantic role labeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 627–637.

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

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

Christiane Fellbaum. 1998. Wordnet: An electronic lexical database cambridge. *MA: MIT Press*.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288.

Seraphina Goldfarb-Tarrant, Tuhin Chakrabarty, Ralph Weischedel, and Nanyun Peng. 2020. Content planning for neural story generation with aristotelian rescoring. *arXiv preprint arXiv:2009.09870*.

Arthur C Graesser, Murray Singer, and Tom Trabasso. 1994. Constructing inferences during narrative text comprehension. *Psychological review*, 101(3):371.

Melanie C. Green and Timothy C. Brock. 2000. The role of transportation in the persuasiveness of public narratives. *Journal of Personality and Social Psychology*, 79(5):701–721.

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

Jena D. Hwang, Chandra Bhagavatula, Ronan Le Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and Yejin Choi. 2021. Comet-atomic 2020: On symbolic and neural commonsense knowledge graphs. In *AAAI*.

Ahmed Khalifa, Gabriella AB Barros, and Julian Togelius. 2017. Deeptingle. *arXiv preprint arXiv:1705.03557*.

Michael Lebowitz. 1987. Planning stories. In *Proceedings of the 9th annual conference of the cognitive science society*, pages 234–242.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. *arXiv preprint arXiv:1707.07045*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Leib Litman, Jonathan Robinson, and Tzvi Abberbock. 2017. Turkprime. com: A versatile crowdsourcing data acquisition platform for the behavioral sciences. *Behavior research methods*, 49(2):433–442.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Lara Martin, Prithviraj Ammanabrolu, Xinyu Wang, William Hancock, Shruti Singh, Brent Harrison, and Mark Riedl. 2018. Event representations for automated story generation with deep neural nets. In *Proceedings of AAAI*, volume 32.

James Richard Meehan. 1976. *The Metanovel: Writing Stories by Computer.* Yale University.

Todor Mihaylov and Anette Frank. 2018. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 821–832.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. In *Proceedings of NAACL-HLT*, pages 839–849.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Xiangyu Peng, S. Li, Spencer Frazier, and Mark O. Riedl. 2020. Reducing non-normative text generation from language models. In *International Conference on Natural Language Generation*.

Xiangyu Peng, Siyan Li, Sarah Wiegreffe, and Mark Riedl. 2021. Inferring the reader: Guiding automated story generation with commonsense reasoning. *arXiv preprint arXiv:2105.01311*.

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

Christopher Purdy, Xinyu Wang, Larry He, and Mark Riedl. 2018. Predicting generated story quality with quantitative measures. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 14.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Hannah Rashkin, Asli Celikyilmaz, Yejin Choi, and Jianfeng Gao. 2020. Plotmachines: Outline-conditioned generation with dynamic plot state tracking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4274–4295.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

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

Melissa Roemmele. 2016. Writing stories with help from recurrent neural networks. In *Proceedings of AAAI*, volume 30.

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

Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. 2019. The woman worked as a babysitter: On biases in language generation. *arXiv preprint arXiv:1909.01326*.

Peng Shi and Jimmy Lin. 2019. Simple bert models for relation extraction and semantic role labeling. *arXiv preprint arXiv:1904.05255*.

Robert Speer and Catherine Havasi. 2013. Conceptnet 5: A large semantic network for relational knowledge. In *The People's Web Meets NLP*, pages 161–176. Springer.

Pradyumna Tambwekar, Murtaza Dhuliawala, Lara J Martin, Animesh Mehta, Brent Harrison, and Mark O Riedl. 2019. Controllable neural story plot generation via reinforcement learning. In *Proceedings of the 28th IJCAI*.

Wilson L Taylor. 1953. "cloze procedure": A new tool for measuring readability. *Journalism quarterly*, 30(4):415–433.

Stephen Ware and Cory Siler. 2021. Sabre: A narrative planner supporting intention and deep theory of mind. In *Proceedings of the 17th AAAI International Conference on Artificial Intelligence and Interactive Digital Entertainment*.

Stephen G Ware and R Michael Young. 2010. Modeling narrative conflict to generate interesting stories. In *Sixth Artificial Intelligence and Interactive Digital Entertainment Conference*.

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

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1097–1100.

Rolf A Zwaan and Gabriel A Radvansky. 1998. Situation models in language comprehension and memory. *Psychological bulletin*, 123(2):162.

A Graph Expansion Details

For current knowledge graph $\mathbf{G}_{ti} \in \mathbf{G}_t$, where $i \in [1, K]$, we expand each concept by m_1 depth. We firstly expand all the physical entities in current knowledge graph $\mathbf{G}_{ti}$:

1. For every physical entity in $\mathbf{G}_{ti}$, we infer as set of relevant entities to construct $E_{ti,\text{entity}}$ from ConceptNet.
2. For each physical entity in $E_{ti,\text{entity}}$ (i.e. k_1th event, $e^{k_1}_{ti,\text{entity}}$, where $k_1 \in [1, n_1]$), keep inferring relevant entities to construct $E^{k_1,2}_{ti,\text{entity}}$ until we get m_1 deep inference, $E^{k_1,m_1}_{ti,\text{entity}}$.
3. Hence, we obtain an inference event set $\widetilde{E^{k_1}_{ti,\text{entity}}} = \{e^1_{ti,\text{entity}}\} \cup E^{k_1,2}_{ti,\text{entity}} \cup \ldots \cup E^{k_1,m_1}_{ti,\text{entity}}$ for entity $e^{k_1}_{ti,\text{entity}}$.

Next, we consider social events:

1. Construct another inference event set $E_{ti,event}$ by inferring social interaction of current story history by COMET$^{20}_{20}$. n_2 is the beam size of the output of COMET$^{20}_{20}$. For example, the `xEffect` of "graduate from college" is "get degree" (green nodes in red bubbles of Fig. 2).
2. For each event in $E_{ti,event}$ (i.e. k_2th event, $e^{k_2}_{ti,event}$), keep inferring social interaction to construct $E^{k_2,2}_{ti,event}$ until we get m_1 level inference, $E^{k_2,m_1}_{ti,event}$. Hence, we obtain inference event set $\hat{E}^{k_2}_{ti,event} = \{e^1_{ti,event}\} \cup E^{k_2,2}_{ti} \cup \ldots \cup E^{k_2,m_1}_{ti,event}$ for $e^{k_2}_{ti,event}$.
3. For each event in new inference event set—$\hat{E}^{k_2}_{ti,event}$, we infer all the relevant entities by COMET$^{20}_{20}$ and ConceptNet to construct $\tilde{E}^{k_2}_{ti,event}$. For example, "get degree" is relevant with "job". Then we obtain inference set $\widetilde{E^{k_2}_{ti,event}} = \hat{E}^{k_2}_{ti,event} \cup \tilde{E}^{k_2}_{ti,event}$ for event $e^{k_2}_{ti,event}$.

Hence, for $\mathbf{G}_{ti}$, we totally obtain n entity candidates $e^k_{ti} \in \{E_{ti,\text{entity}} \cup E_{ti,event}\}$, where $k \in [1, n]$ and $n = n_1 + n_2$ and its corresponding knowledge graph candidates $\mathbf{G}'_{ti} = \{\mathbf{G}'^1_{ti}, \ldots, \mathbf{G}'^n_{ti}\}$, where $\mathbf{G}'^k_{ti} = \mathbf{G}_{ti} \cup \widetilde{E^k_{ti}}$, where $\widetilde{E^k_{ti}} \subset \{E^{k_1}_{ti,\text{entity}} \cup \widetilde{E^{k_2}_{ti,event}}\}$. Totally, we obtain $\mathbf{G}'_t = \{\mathbf{G}'_{t1}, \ldots, \mathbf{G}'_{tK}\}$.

We also expand the depth of the graph for goals by m_2 via similar means:

1. For each node in $\mathbf{G}_{\text{goal}}$, we expand all the relevant entities with Conceptnet by m_2 depth to construct $\widetilde{E_{\text{goal,enitity}}}$.
2. Construct event inference set E^1_{goal} by inferring social interaction of story goal by COMET$^{20}_{20}$. For example, the `xNeed` of "enjoy sunshine" is "go to beach" (green nodes in Fig. 2).
3. Keep inferring social interaction of all the events in E^n_{goal} to construct E^{n+1}_{goal} until we get $E^{m_2}_{\text{goal}}$. $\hat{E}_{\text{goal}} = \{E^1_{\text{goal}}, \ldots, E^{m_2}_{\text{goal}}\}$.
4. For each event in $\hat{E}_{\text{goal}}$, and all the nodes in $\mathbf{G}_{\text{goal}}$, we infer all the relevant entities to construct $\tilde{E}_{\text{goal}}$ by COMET$^{20}_{20}$ and ConceptNet. $\widetilde{E_{\text{goal,event}}} = \hat{E}_{\text{goal}} \cup \tilde{E}_{\text{goal}}$.
5. Obtain the updated knowledge graph $\mathbf{G}'_{goal} = \mathbf{G}_{\text{goal}} \cup \widetilde{E_{\text{goal,event}}} \cup \widetilde{E_{\text{goal,enitity}}}$.

B Implementation Details

B.1 Semantic Role Labeling Using VerbAtlas

The SRL model provides the automatic identification and labeling of argument structures of stories. For example, it extracts `'verbatlas': 'EXIST_LIVE', 'args_words': {'Theme': 'Jenny', 'Attribute': 'Florida'}` from "*Jenny lived in Florida*". Verbs in the story will be represented as the VerbAtlas frame. For example, **"`live`"** is represented as **"`EXIST_LIVE`"**.

For the semantic role labeling model (SRL), we use a fine-tuned transformer model proposed by (Shi and Lin, 2019) which is the current state-of-the-art for English SRL. It is a BERT (Devlin et al., 2019) model with a linear classification layer trained on the Ontonotes 5.0 dataset to predict PropBank SRL. We use an open-source implementation [5], which is based on the official AllenNLP BERT-SRL model [6]. Trained with the following hyperparameters:

- Batch size: 32
- Dropout for the input embeddings: 0.1
- Learning rate: $5e^{-5}$
- Optimizer: Adam
- Total Epochs: 15

Then, we use the mappings from Propbank frames to VerbAtlas (Di Fabio et al., 2019) classes to return the correct corresponding VerbAtlas classes instead of Propbank's (Palmer et al., 2005). The direct mapping is possible because, for every VerbAtlas class, there is only one ProbBank frame, which allows us to utilize the rich content provided

[5] https://github.com/Riccorl/transformer-srl
[6] https://demo.allennlp.org/semantic-role-labeling

Type	Definition
AtLocation	located or found at/in/on
CapableOf	is/are capable of
HasA	has, possesses or contains
HasProperty	can be characterized by being/having
MadeOf	is made of
MadeUpOf	made (up) of
MotivatedByGoal	is a step towards accomplishing the goal
UsedFor	used for
PartOf	is a part of

Table 3: Definitions of the selected types of ATOMIC relations that we used for graph expansion.

Type	Definition
xWant	as a result, PersonX wants
xNeed	before this event, PersonX needed
xEffect	as a result, PersonX will
oWant	as a result, PersonY or others want
oEffect	as a result, PersonY or others will

Table 4: Definitions of the selected types of ATOMIC relations that we used for graph expansion.

by VerbAtlas while using the same model initially trained to predict ProbBank.

B.2 Common-sense Inference Types

Table 3 shows the inference relations of ATOMIC we use for inferring physical entity. Table 4 shows the inference relations of ATOMIC we use for inferring events.

B.3 Continuation Candidate Generation Details

We first use pattern package[7] to extract tense of prompt and run verb conjugation for each event e^k_{ti}. Then we use nltk toolkit[8] to extract adjective and noun set from e^k_{ti}. Each event e^k_{ti} is represented as three sets—verb, noun and adjective. For example, when prompt is "Jenny celebrated her birthday.", the event "get a gift" is split into three sets $\{\{"got"\}, \{gift\}, \{\}\}$. When prompt is "Jenny eats a lot of ice cream.", event "get fat" is split into three sets $\{\{"gets"\}, \{\}, \{fat\}\}$. We generate templates with the following rules,

- Before the subject token, we put $0 \sim 5$ $\langle mask \rangle$ tokens.
- Subject tokens are (1) Previous occurred subjects, i.e. *"Jenny"*, *"Bob"*; (2) $\langle mask \rangle$.
- Between the subject token and the verb token, we put $0 \sim 2$ $\langle mask \rangle$.
- Between the verb token and the adjective token, we put $0 \sim 2$ $\langle mask \rangle$.
- Between the adjective token and the noun token, we put $0 \sim 2$ $\langle mask \rangle$.
- After the noun token, we put $0 \sim 8$ $\langle mask \rangle$ tokens.

Examples (we show three) are as follows when e^k_{ti} ="`go to beach`" and prompt is "Jenny lived in Florida.",

- $\langle mask \rangle$ `Jenny went` $\langle mask \rangle$ `beach` $\langle mask \rangle$.
- `Jenny` $\langle mask \rangle$ `went` $\langle mask \rangle$ `beach`.
- `Jenny` $\langle mask \rangle$ $\langle mask \rangle$ `went beach` $\langle mask \rangle$ $\langle mask \rangle$ $\langle mask \rangle$.

B.4 RoBERTa Fine-tuning

We fine-tune RoBERTa-large (Liu et al., 2019) on ROCStories (Mostafazadeh et al., 2016), Writing Prompts(Fan et al., 2018), Fairytale stories(Ammanabrolu et al., 2020b) separately to infill the mask tokens in the given text template. We pre-process the datasets by masking 15% of all the tokens randomly, concatenating all texts together, and splitting them into chunks of the same length (equal to 128). Each chunk is then used as one training sample.

During fine-tuning, we use the AdamW optimizer (Loshchilov and Hutter, 2017) to train the RoBERTa for 3 epochs with batch size = 8. Other optimizer-related hyperparameters are attached as follows.

- learning rate: $\gamma = 2 \times 10^{-5}$
- betas: $\beta_1 = 0.9, \beta_2 = 0.999$
- epsilon: $\epsilon = 10^{-8}$
- weight decay: $\lambda = 0.01$

B.5 Baselines—C2PO(Ammanabrolu et al., 2020a)

We replicate the C2PO model by Ammanabrolu et al. using the code published on the paper's public repository. [9] All the encoder and model checkpoints are provided by the author.

B.6 Baselines— Goldfarb-Tarrant et al. (2020)

We firstly extract high-level outlines with the help of codes provided by Ammanabrolu et al. (2020a)[10]. Then we split the long Writing Prompt stories into sections according to extracted

[7] https://github.com/clips/pattern
[8] https://www.nltk.org/
[9] https://github.com/rajammanabrolu/C2PO
[10] https://github.com/rajammanabrolu/C2PO/tree/master/Plot-Extraction

plots. Each high-level plot is used as prompt and section in between as story. We fine-tune BART model using the code and parameters published on the paper's public repository. [11]

[11] https://github.com/PlusLabNLP/story-gen-BART

Precision %	Recall %	# of triplets
$81.96^{\ddagger}$	$72.89^{\ddagger}$	255

Table 5: Results of evaluating knowledge graph triplets. ‡ indicates $\kappa > 0.4$ or moderate agreement.

C Knowledge Graph Acquisition Evaluation

We assess whether knowledge graph can acquire the story world state accurately and comprehensively. We randomly select 125 sentences from ROCStories and convert them into knowledge graph triples. Human participants were asked to validate each graph triples given the sentence and then write down the missing information. For example, they need to check whether $\langle jenny, LIKE, beach \rangle$ given "*Jenny likes beach and sunshine*" is correct and write down the missing concept, "`sunshine`". The detail of this study is shown in Appendix E.1.

Table 5 shows the accuracy (precision) and sensitivity (recall) of the extracted knowledge graph triples. We treat the majority vote from human participants as the ground-truth. *Precision* is the fraction of extracted triples that are correct rated by human participants. *Recall* is the fraction of the triples that are successfully extracted from stories. Precision, 81.96%, shows that the knowledge graph can represent the information in sentences accurately. Recall, 72.89%, proved that the knowledge graph can represent most of the information in sentences. Both of these two metrics have moderate agreement. This indicates that the knowledge graph extracted from sentences matches reader expectations and can be used as story world state upon which to base further story generation.

D Experiments Details

ROCStories. We firstly compare our model with C2PO on ROCStories. Since the length of ROCStories is 5 sentences. We randomly select 15 stories and use the first sentence as prompt and the last sentence as goal to generate stories.

StoRM firstly convert first and last sentence as story KG and goal KG. Then we look ahead 2 step ($m_1 = m_2 = 2$ in Section 3.2) and set the max story length as 4. When StoRM finds a inference link to reach the goal or the graph similarity is over 80%, the system will **early stop**. Beam size of COMET is set as 5. Hence the story length of StoRM is $2 \sim 6$ (prompt and goal sentences are counted).

C2PO finds a inference link between the first sentence and the last sentence by generate 3 event candidates twice going forward and backward, respectively. Hence the story length of StoRM is $2 \sim 6$ (prompt and goal sentences are counted). Examples can be found in Table 6.

Writing Prompts We count the length of Writing Prompts by splitting stories with period, question mark and exclamation mark. To use StoRM to generate stories on Writing Prompts, we first follow Ammanabrolu et al. (2020a) to extract high-level plots from Writing Prompts stories. Then we use the first event in the extracted plots as prompt to start StoRM and Goldfarb-Tarrant et al. (2020). Goal is the second event of extracted plots.[12] We look ahead 2 step ($m_1 = m_2 = 2$ in Section 3.2) and set the max story length as 5. When StoRM finds a inference link to reach the goal or the graph similarity is over 80%, or 5 sentences have been generated, the system will **early stop**. Beam size of COMET is set as 5. Hence the local story length of StoRM is $2 \sim 6$(prompt and goal sentences are counted). We then append the goal to the story history, feed the third plot as goal and keep generation with the same process until we use all the plots up.

To seed the Goldfarb-Tarrant et al. (2020), we use the same high-level plots as prompts. Then we truncate the generated stories to 6 sentences and append the next plot to story history. We then keep generating story on the next plot until we use all the plots up. Totally we randomly select 15 generated stories[13] for human evaluation. More examples can be found in Table 7.

[12]Goals are only provided to StoRM, because Goldfarb-Tarrant et al. (2020) is unconstrained story generator.

[13]Each story is generated on one plot and its next plot is used as the goal.

StoRM: **Justin decided to make dinner for his boyfriend.** So he tried to cook spaghetti. But, he failed. In fact, he ate a lot of spaghetti everyday. So he **ordered pizza** instead.
C2PO: **Justin decided to make dinner for his boyfriend.** Justin tries to cook. Justin wants clean up. Justin tries to work. Justin starts to have money. **Justin ordered pizza.**
StoRM: **Robin was afraid of flying.** The first time he flew, it scared him. After that he got more nervous. Now all he wanted was to calm his down. He took a deep breath. Robin **felt less nervous** now.
C2PO: **Robin was afraid of flying.** Robin wants to save himself. Robin wants to find a job. Robin tries to have a job. Robin wants to be nervous. **Robin felt a lot less nervous about her flight.**
StoRM: **Anna was having a bad day at work.** She wished she had stayed home. She was glad that friends invited her to movie night. She had a fun time with them. She is **grateful and pleased** now.
C2PO: **Anna was having a bad day at work.** Anna tries to get some rest. Anna wants to get up. Anna begins go to the store. Anna starts to have received a gift. **Anna was grateful and pleased.**
StoRM: **David noticed he had put on a lot of weight recently.** He starts to eat less. So he exercised more every day. And he **felt much better** afterwards.
C2PO: **David noticed he had put on a lot of weight recently.** David starts to eat less. David begins to exercise. David begins to rest. **He started to feel much better.**

Table 6: Story examples generated by StoRM and C2PO on ROCStories. Bolded words are prompts and goals. The goal of StoRM is knowledge graph, so it will generated the goal sentence by itself if StoRMfinds a inference link to reach the goal or similarity with goal knowledge graph is over threshold. C2PO will use given goal sentence in the generated stories directly.

StoRM: **Sherlock sat in the downstairs sitting room.** When he talked to people in the room. he never got a response back. So he thought about what was wrong. **Sherlock himself stared into space , deep in thought .** In fact, he never thought of anyone. He thought that he was alone in the world. Then, he found his mate. There, he got a hug from his wife. He asked if he could put his coat on her. **He straightened up and donned his favourite coat.**
Goldfarb-Tarrant et al. (2020) **Sherlock sat in the downstairs sitting room.** The room was silent, save for a man sitting in his own chair. the chair made of wood, built of oak. a chair that would make a good bed and a bed that would be comfortable. Looking around at the room, the woman saw a small smile cross Holmes ' face, but she did n't say anything. **Sherlock himself stared into space, deep in thought.** He was cold. The cold. He felt the cold of the night air around him. His eyes looked into the darkness that seemed so much more than the cold and he sat there. The cold took his in. It was cold, and it seemed like forever as his eyes stared into the dark that seemed to be the end. **He straightened up and donned his favourite coat.**
StoRM: **Sarah sat with her dog.** she had a companion. her dog. One day she took the animal out with her. They enjoyed the sunshine and weekend. She whispered soothing words into its ear. **She tried talking to the dog while she pet it.** Suddenly, she was bited by the dog. She then yelled at the dog. She wanted to get the dog to stop. She finally got the dog to stop. **The dog made cute noises.**
Goldfarb-Tarrant et al. (2020) **Sarah sat with her dog**. The girl was young and pretty, but she was also scared. Her mother stood before her, and the woman seemed like hers to her. " Sarah. " The girl looked back and forth between them, and the woman, and then his eyes looked at him for a moment, and all she could see was fear, and sadness. Sarah looked away. The man, who was older than the two of herself, and had a dark hairline and dark eyes. **She tried talking to the dog while she pet it.** My mind raced. It was the same as it always had. I was the only one in the room and he knew what he wanted, a man who wanted my love, my life. A man who was so perfect for me. The man came to mine in our bedroom, we would go out for a drink. **The dog made cute noises.**

Table 7: Story examples generated by StoRM and C2PO. Bolded words are prompts and goals.

StoRM: **Bearskin clip his nails.** When he just got a haircut , his hair looks great. He thought he **looked good** in it. In fact , he has won every competition in the years. In fact , he has got every trophy in the years. However , he never showed it. In fact , he always hid from everyone. When he met other people , he was shy. **Bearskin dropped his half of ring.**Then , he picked the ring it up. Then , he proposed to her with the ring. **He was her bridegroom.**
C2PO: **Bearskin clip his nails.** Bearskin tries to clean them. Bearskin begins wash hands. Bearskin tries to go to school. Bearskin starts to train consistently. **Bearskin is good.** Bearskin wants to win. Bearskin wants to celebrate. Bearskin wants to get a ring. **Bearskin dropped his half of ring.** Bearskin begins to get a new ring.Bearskin tries buy a ring. Bearskin wants to have a wedding. **He was her bridegroom.**
StoRM: **Girl did hard work.** When she was promoted to the new job. **Girl cried.** She met with a boy. She dated with him for one year. **Girl got a ring from him.** Then , she got to kiss the boy. Finally , she had sex with the boy. **She had a baby.**
C2PO: **Girl did hard work.** Girl tries to celebrate. Girl tries to go to bed. Girl wants to fall. Girl begins to get hurt. **Girl cried to God.** Girl tries to feel better. Girl starts to go to the bathroom. Girl wants to get ready. Girl begins to go to the ring. **Girl got a ring.** Girl begins to buy it. Girl wants to open it. Girl tries to open the door. Girl starts to bring the baby to the car. **She had her baby.**
StoRM: **Old soldier returned from war.** Then he found his family. Then he told his family what happened. He said he was yelled at. Then he got very angry and hurt others. Then he was arrested. When he went to court , **he met the the woman** in black. He met with woman. he never had a relationship with the woman. He began to date with her. He could not move his eyes away from her. **He observe king 's daughters.**
C2PO: **Old soldier returned from war.** Old soldier wants to get a drink. Old soldier starts to go out. Old soldier starts to meet the woman. **He met with woman.** Old soldier wants to get to know them. Old soldier wants to ask them out. Old soldier starts to get married. Old soldier begins to get married. **He observe king 's daughters.**

Table 8: Story examples generated by StoRM and C2PO on Fairytale Stories. Bolded words are prompts and goals. The goal of StoRM is knowledge graph, so it will generated the goal sentence by itself if StoRMfinds a inference link to reach the goal or similarity with goal knowledge graph is over threshold. C2PO will use given goal sentence in the generated stories directly.

Fairytale stories We first follow Ammanabrolu et al. (2020a) to extract high-level plots from fairytale stories. We use the first plot in the extracted plots as prompt to start StoRM and C2PO. Goal is the second plot of extracted plots. StoRM firstly convert first plot and second plot as story KG and goal KG. Then we look ahead 2 step ($m_1 = m_2 = 2$ in Section 3.2) and set the max story length as 4. When StoRM finds a inference link to reach the goal or the graph similarity is over 80%, the system will **early stop**. Beam size of COMET is set as 5. Hence the local story length of StoRM is $2 \sim 6$ (prompt and goal sentences are counted). StoRM keeps generating story on the previous story and set goal as the next plot of

extracted plots until we use all the plots up.

We follow the original paper, we ask C2PO to fill in the section between high-level plots with the inference link it finds as generated stories. Totally we randomly select 15 generated stories[14] for human evaluation. More examples can be found in Table 8.

[14]Each story is generated on one plot and its next plot is used as the goal.

E Human Evaluation Details

E.1 Knowledge Graph Acquisition Evaluation Set-up

We ask participants a set of screen questions to make sure they understand our task. The details can be found in Figure 4. We conduct our studies using the Cloud Research crowdsourcing platform to interact with Amazon Mechanical Turk (Litman et al., 2017). Obtaining at least a bachelor's degree and English as their native language are required to take this study. Participants are required to pass screening questions and then explain their preferences of each choice in this study with more than 50 characters, which helps filter out low-quality responses and ensures the validity of the study. Our study was approved by our Institutional Review Board, and we payed participants the equivalent of $15/hr.

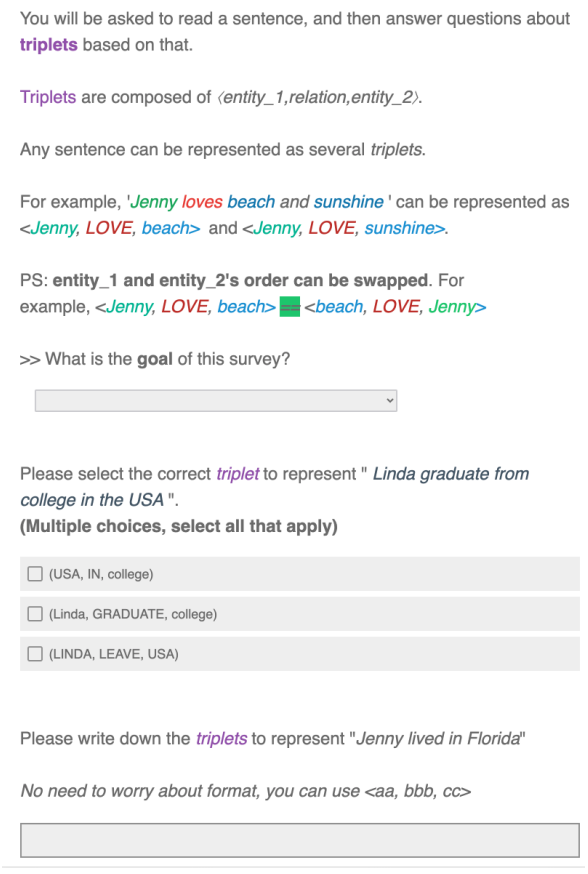

Figure 4: Screenshot of the human study instruction.

We assess whether knowledge graph can acquire the story world state accurately and comprehensively. We randomly select 125 sentences from ROCStories and convert them into knowledge graph triplets. We recruited 30 participants on a crowdsourcing platform. Each participant read a randomly selected subset of knowledge graph triplets (20 sentences per participant). They were asked to validate each graph triplets given the sentence and then write down the missing information. An example is shown in Figure 5. At least 3 crowd workers validate each triple and we take the majority vote as the result.

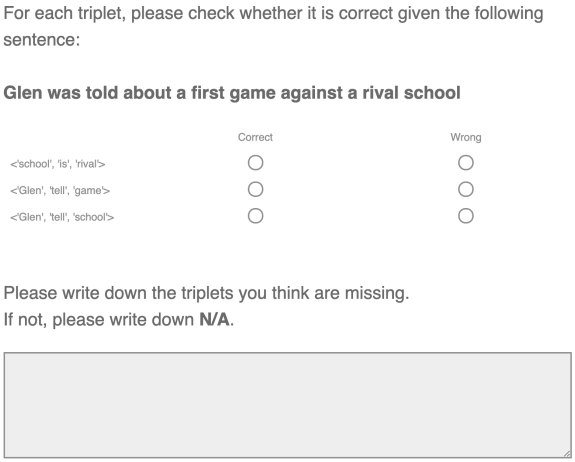

Figure 5: Screenshot of Knowledge Graph Acquisition evaluation.

E.2 Story Coherence Evaluation Set-up

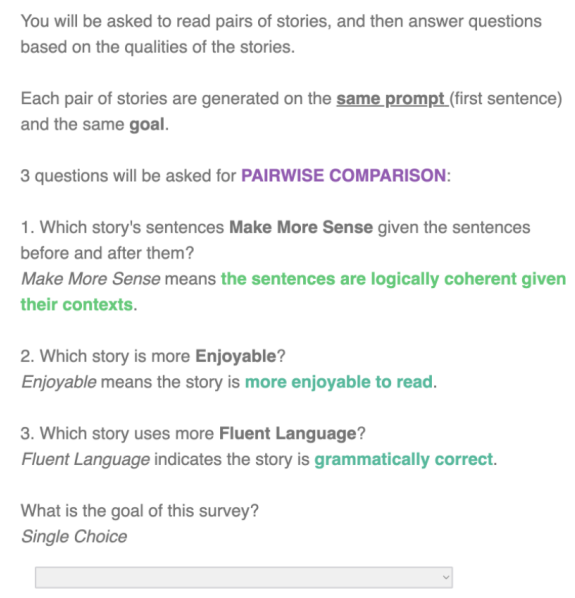

Figure 6: Screenshot of the human study instruction.

We evaluate coherence using human participant evaluation, asking a set of questions that includes dimensions such a logical coherence, loyalty to plot, and enjoyability. We recruited 50 participants on a crowdsourcing platform. We first show them the instruction(Figure 6) and the screen questions

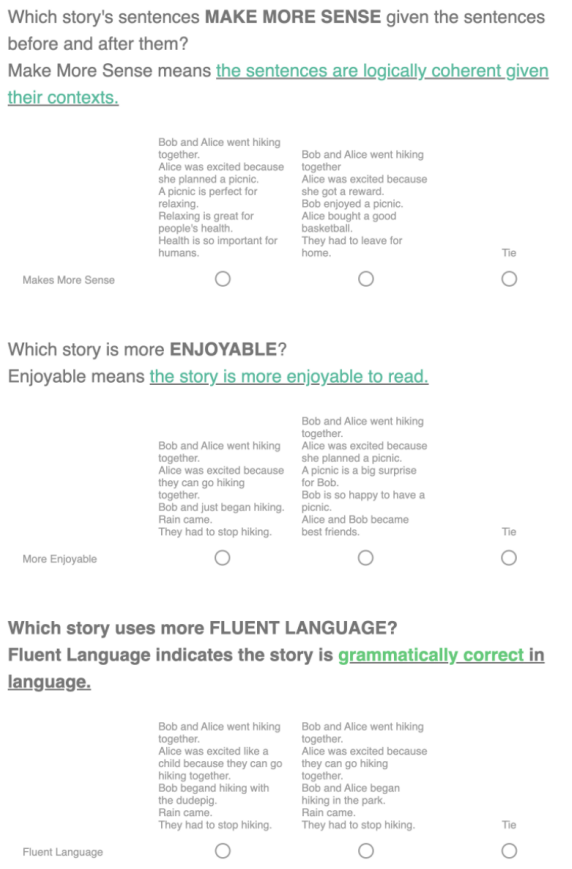

Figure 7: Screenshot of the human study screen questions.

(Figure 7). We then ask questions in Section 4.1 and example is shown in Figure 8.

We conduct our studies using the Cloud Research crowdsourcing platform to interact with Amazon Mechanical Turk (Litman et al., 2017). Obtaining at least a bachelor's degree and English as their native language are required to take this study. Participants are required to pass screening questions and then explain their preferences of each choice in this study with more than 50 characters, which helps filter out low-quality responses and ensures the validity of the study. Our study was approved by our Institutional Review Board, and we payed participants the equivalent of $15/hr.

E.3 Controllability Evaluation Set-up

We recruited 48 participants on a crowdsourcing platform. We firstly show human participants instructions (Figure 9) and screen questions (Figure 10) to make sure they understand the task. We then ask them to read a randomly selected generated story pairs—one from StoRM and one from our baselines. They then answered which one better

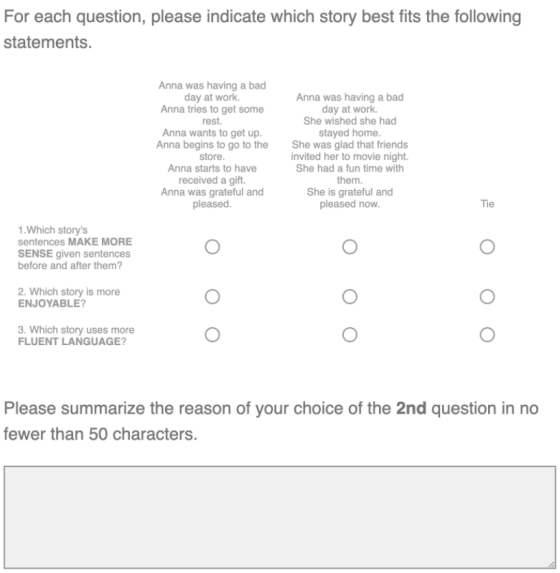

Figure 8: Screenshot of the human study on evaluating coherence.

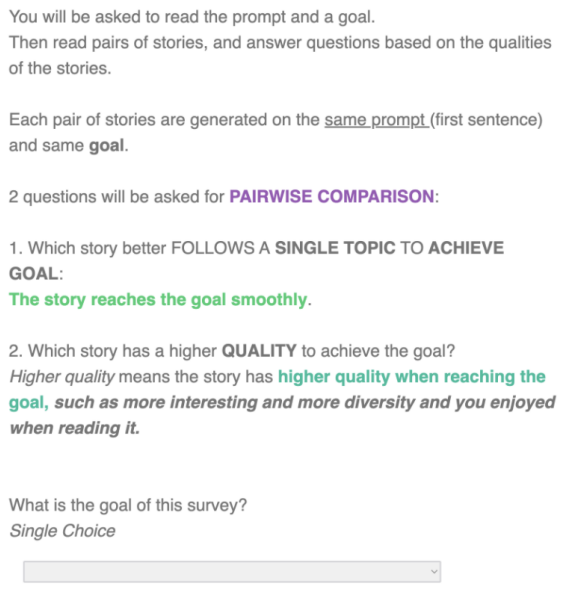

Figure 9: Screenshot of the instruction on evaluating coherence.

met the criteria in Section 4.2. Example is shown in Figure 11.

We conduct our studies using the Cloud Research crowdsourcing platform to interact with Amazon Mechanical Turk (Litman et al., 2017). Obtaining at least a bachelor's degree and English as their native language are required to take this study. Participants are required to pass screening questions and then explain their preferences of each choice in this study with more than 50 characters, which helps filter out low-quality responses and

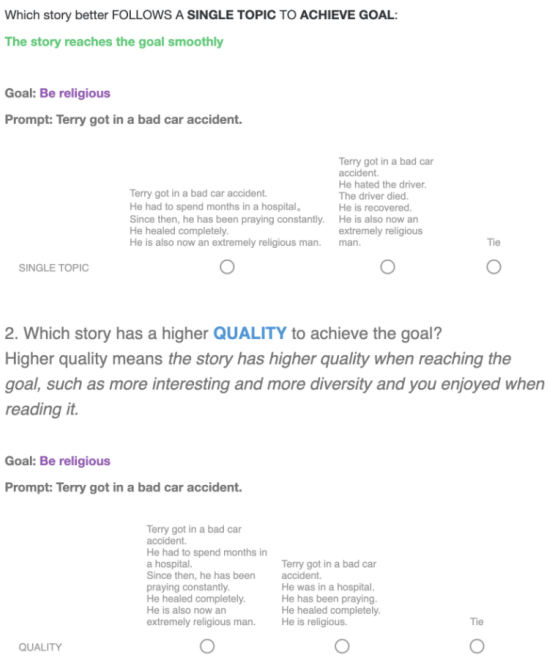

Figure 10: Screenshot of the screen questions on evaluating coherence.

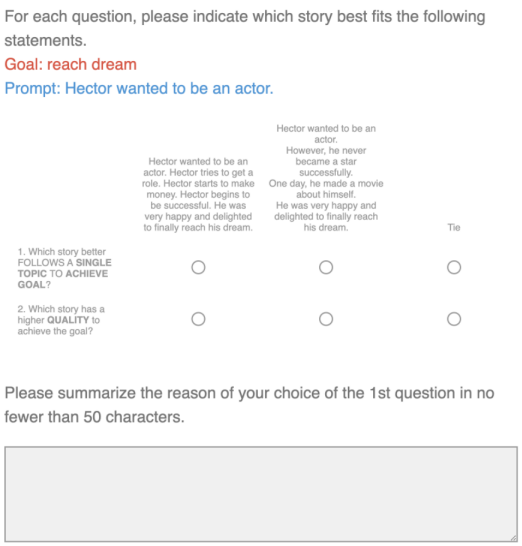

Figure 11: Screenshot of the human study on evaluating coherence.

ensures the validity of the study. Our study was approved by our Institutional Review Board, and we payed participants the equivalent of $15/hr.

F Evaluation Results

F.1 Story Coherence Evaluation

We also report the majority vote and agreement results in Table 9. We additionally observe that these three metrics are positively correlated using Spearman's Rank Order Correlation in all of these ablation studies. $r_s = 0.49$, $p < 0.01$, between "Logical Sense" and "Enjoyable"; $r_s = 0.39$, $p < 0.01$, between "Logical Sense" and "Fluency" for comparing to C2PO on ROCStories. $r_s = 0.32$, $p < 0.01$, between "Logical Sense" and "Enjoyable"; $r_s = 0.42$, $p < 0.01$, between "Logical Sense" and "Fluency" and $r_s = 0.32$, $p < 0.01$, between "Enjoyable" and "Fluency" for comparing to Goldfarb-Tarrant et al. on Writing Prompts. $r_s = 0.52$, $p < 0.01$, between "Logical Sense" and "Enjoyable"; $r_s = 0.49$, $p < 0.01$, between "Logical Sense" and "Fluency" and $r_s = 0.37$, $p < 0.01$, between "Enjoyable" and "Fluency" for comparing to C2PO on fairy tale story dataset.

F.2 Story Controllability Evaluation

We also report the majority vote and agreement results in Table 10. We additionally observe that these two metrics are positively correlated using Spearman's Rank Order Correlation in all of these ablation studies. $r_s = 0.42$, $p < 0.01$, between "goal" and "quality" for comparing to C2PO on ROCStories; $r_s = 0.39$, $p < 0.01$, between "goal" and "quality" for comparing to Goldfarb-Tarrant et al. on Writing Prompts; and $r_s = 0.39$, $p < 0.01$, between "goal" and "quality" for comparing to C2PO on fairy tale stories.

Models	Data set	Logical Sense			Enjoyable			Fluency		
		Win%	Lose%	Tie%	Win%	Lose%	Tie%	Win%	Lose%	Tie%
StoRM vs CP	WP	**86.6****†	6.7	6.7	**60.0**	40.0	0.0	**80.0****	**10.0**	10.0
StoRM vs C2PO	ROC	**80.0****†	10.0	10,0	**80.0****	10.0	10.0	**60,0****†	**0,0**	40.0
	FT	**60.0***	33.3	6.7	**73.3****	16.7	10.0	**60.0**†	20.0	20.0

Table 9: Coherence evaluation results, showing the majority vote of participants who preferred the first system, second system, or thought the systems were equal. CP indicates (Goldfarb-Tarrant et al., 2020). Each system is conditioned on the same test-set prompts and same goal. * indicates results are significant at $p < 0.05$ confidence level; ** at $p < 0.01$ using a Wilcoxan sign test on win-lose pairs. † indicates $\kappa > 0.2$ or fair agreement.

Models	Data set	Goal			Quality		
		Win%	Lose%	Tie%	Win%	Lose%	Tie%
StoRM vs CP	WP	**60.0***†	6.7	33.3	**53.3****	0.0	46.7
StoRM vs C2PO	ROC	**86.6****†	6.7	6.7	**86.7****	0.0	13.3
	FT	**60.0***	6.7	33.3	**60.0***	6.7	33.3

Table 10: Controllability evaluation results, showing the majority vote of participants who preferred the first system, second system, or thought the systems were equal. CP indicates Goldfarb-Tarrant et al.(2020). Each system is conditioned on the same test-set prompts. * indicates results are significant at $p < 0.05$ confidence level; ** at $p < 0.01$ using a Wilcoxan sign test on win-lose pairs. † indicates $\kappa > 0.2$ or fair agreement.

G Ablation Study

We perform ablation studies to choose the best hyperparameters, We build goal story world states (§3.1) on the last sentence of randomly selected 30 stories from ROCStories (Mostafazadeh et al., 2016) to guide the story generation process. StoRM keeps generating story continuations until knowledge graph difference score $R(s)$ reaches 0.8 or generated story hits the goal ($r_1(s) = 1$). We measure the following metric:

- *Average story length* (Avg. len): Calculate the average story length which is required to reach $R(s) = 0.8$ (§3.3) or generated story hits the goal ($r_1(s) = 1$). Smaller average story length stands for faster, and thus more direct, goal achievement. Because the system finds different ways to achieve the goal. As with many planning systems, ours has a bias toward shorter, more compact solutions. We stop generation when story length reaches 10.

Model		Avg. len ↓
StoRM Full	$\alpha = 0.50$	**6.92** $\pm$ 1.21
	$\alpha = 0.90$	9.27 $\pm$ 1.50
	$\alpha = 0.25$	8.32 $\pm$ 0.94

Table 11: Results of the ablation study. α is tuning the inference contribution when calculating graph difference.

Table 11 shows the result of the ablation study. We experiment with three values of α in our StoRM framework. The best performing model has $\alpha = 0.5$, balancing between inference-node-guided and goal-node-guided story generation, where larger α indicates more inference-node-driven.