Inferring the Reader: Guiding Automated Story Generation with Commonsense Reasoning

Xiangyu Peng♣∗ Siyan Li♣∗ Sarah Wiegreffe† Mark Riedl♣

Georgia Institute of Technology
†Allen Institute for Artificial Intelligence
{xpeng62, sl1613}@gatech.edu
wiegreffesarah@gmail.com
riedl@cc.gatech.edu

Abstract

Transformer-based language model approaches to automated story generation currently provide state-of-the-art results. However, they still suffer from plot incoherence when generating narratives over time, and critically lack basic commonsense reasoning. Furthermore, existing methods generally focus only on single-character stories, or fail to track characters at all. To improve the coherence of generated narratives and to expand the scope of character-centric narrative generation, we introduce Commonsense-inference Augmented StoryTelling (CAST), 1 a framework for introducing commonsense reasoning into the generation process with the option to model the interaction between multiple characters. We find that our CAST method produces significantly more coherent, on-topic, enjoyable and fluent stories than existing models in both the single-character and two-character settings in three storytelling domains.

1 Introduction

AI storytelling is a crucial component of computational creativity. Humans use storytelling to entertain, share experiences, educate, and to facilitate social bonding (Riedl and Young, 2010). For an intelligent system to be unable to generate a story limits its ability to interact with humans in naturalistic ways (Riedl, 2016). Automated Story Generation, the task of requiring a system to construct a sequence of sentences that can be read and understood as a story, is a grand challenge in AI.

Prior to the advent of neural language models, methods to model the narrative arcs of stories leveraged a variety of statistical techniques to track events and characters (Gervás, 2013, 2014; Ouyang and McKeown, 2015). The dominant approach to story generation today is to use neural language models (Roemmele, 2016; Khalifa et al., 2017; Clark et al., 2018; Martin et al., 2018). When a language model is trained on a corpus of stories, samples from the resulting distribution tend to also be stories. These techniques have improved with the adoption of Transformer-based models, such as GPT-2 (Radford et al., 2019) and GPT-3 (Brown et al., 2020). However, these models are prone to generating repetitive or generic continuations (Holtzman et al., 2019). Furthermore, as the length of the story grows, these models can lose coherence. Other artifacts include new characters being arbitrarily introduced at any time and characters being forgotten. One reason for these phenomena is that language models generate continuations by sampling from a learned distribution $P_θ(tok_n|tok_{<n})$. Human readers, however, do not
perceive the coherence of a narrative as a function of the likelihood of seeing specific continuations of previous contexts. Statistical sampling from a distribution is not constrained to making logical transitions because the rich relationships that readers make to perceive coherence are not modeled.

Previous attempts to enhance story generation coherence use conditioning on content-relevant features such as plot outlines (Fan et al., 2018; Peng et al., 2018; Rashkin et al., 2020), or character emotional arcs (Brahman and Chaturvedi, 2020). These improve plot coherence through adherence to a manually-given high-level plan. A high level plan can also be automatically generated then decomposed (Yao et al., 2019; Fan et al., 2019; Ammanabrolu et al., 2020b), which elevates the challenges of maintaining coherence to a higher level of abstraction. Neural language models can also be fine-tuned on other signals such as commonsense knowledge or progression rewards (Guan et al., 2020; Tambwekar et al., 2019), which improves the distribution but still relies solely on sampling and the assumption that language models can encode complex story structure in token distributions.

The latent state of neural language models used to generate subsequent story continuations are unlikely to relate to a human reader’s mental model of the state of a story world. Studies of human reader comprehension (Trabasso and Van Den Broek, 1985; Graesser et al., 1991, 1994) show that readers comprehend stories by tracking the relations between events. Specifically, reader comprehension relies on the tracking of at least four types of relations between events: (1) causal consequence, (2) goal hierarchies, (3) goal initiation, and (4) character intentions. The perceived coherence of a story is thus a function of the reader being able to comprehend how events correlate to each other causally or how they follow characters’ pursuits of implicit goals. We hypothesize that a story generation system that makes decisions on how to continue a story based on tracking and reasoning about events will generate more coherent stories.

Unfortunately, stories don’t always explicitly declare the causal consequences of events or the goals and intentions of characters. That is, sentences describing character actions or external events are rarely explicitly annotated with the characters’ motivations and goals. Readers must infer the characters’ goals, the relationship between their actions and those goals, and how their goals change as a result of the events in their world. The ability to use basic knowledge about goals and about world states falls within the study of commonsense inference. Initial work in this area was limited to modeling dimensions of the “naive psychology” of characters: motivations and emotional reactions (Rashkin et al., 2018). This was later extended to more attributes—ATOMIC (Sap et al., 2019) and ATOMIC20 (Hwang et al., 2021) are event-centric commonsense knowledge bases that contain logical relationships between events and the mental states and attributes of their participants, represented as typed if-then relations. The former contains nine such dimensions, and the latter 23. COMET20 (Hwang et al., 2021), an extension of COMET (Bosselut et al., 2019), is a transformer-based generative model trained on triples from ATOMIC20. Given a sentence, COMET20 infers commonsense attributes about the characters that fall into three categories: (1) social interactions, (2) physical entities, and (3) effect of events inferred from the sentence. We hypothesize that a neural language generator informed about COMET-inferred event effects as well as character intentions and goals can generate more coherent narratives.

To this end, we introduce Commonsense inference Augmented neural StoryTelling (CAST), which infers the causal relations between events as well as the intents and motivations of characters in the story so far in order to generate story continuations that are more coherent to readers. CAST is a straightforward, cognitively inspired method to scaffold the generation of story text when sampling from a language model. By chaining sentence-level COMET inferences to track important implicit elements of the story over time, CAST is able to make more informed choices when sampling story continuations from a neural language model of choice (GPT-2 in our experiments). It can be used to produce both single-character and multiple-character stories. We hypothesize that stricter, more explicit constraints during generation should result in more coherent narratives than generating via sampling from a distribution alone, even if the distribution is fine-tuned. An overview of our method is presented in Figure 1.

To evaluate the efficacy of our proposed method, we conduct a series of human-participant experiments that measure perceptions of logical coherence of CAST against three strong neural language model story generators on three different story cor-
por. Results indicate that the CAST method produces significantly more coherent, on-topic, enjoyable and fluent stories in both the single-character and two-character settings. This result holds even in a genre with a very different type of common-sense than that which COMET is trained on (fairy tales), indicating our method’s generality.

2 Related Work

In addition to the work mentioned in the introduction, we provide a detailed background on story generation systems that emphasize commonsense reasoning and other related techniques. Guan et al. (2019) were the first to propose to incorporate a commonsense knowledge base into the story generation pipeline. Guan et al. (2020) improved upon this method by using the ATOMIC dataset to fine-tune GPT-2, and then fine-tuning a second time on the ROCStories corpus (Mostafazadeh et al., 2016). This system used multi-task learning during a second fine-tuning stage with an auxiliary objective to distinguish true and engineered false stories.

Similarly, Paul and Frank (2021) fine-tune GPT-2 on ROCStories to obey coherence rules generated by separately trained models. At inference-time, the story generation model is fed the first two and the last sentences of ROCStories test instances, making this an infilling rather than open-ended generation task. Brahman and Chaturvedi (2020) fine-tune GPT-2 to generate stories that follow a given emotional arc for a protagonist, using COMET to infer the protagonist’s emotions as labels for their training dataset. They assume five emotions (anger, fear, joy, sadness, neutral) limited to two changes throughout the story, associated with the xReact and oReact inferences produced by COMET. We do not assume a fixed set of commonsense inference values, and we assume a character’s state may change at each new sentence.

The C2PO system (Ammanabrolu et al., 2021) uses COMET to generate successor and predecessor events instead of a language model, performing a bi-directional search from a given start event and a given end event. C2PO assembles the narrative directly from the short, templated sentences produced by COMET. It also assumes the end of the story is known in advance. Like Brahman and Chaturvedi (2020), it focuses on only two dimensions of COMET and only works for a single character. Our work models interactions between multiple characters and takes advantage of a richer set of inferences that COMET and COMET provide, better aligning with the four types of relations key to reader comprehension (Trabasso and Van Den Broek, 1985; Graesser et al., 1991, 1994).

Very recently, Lin et al. (2022) utilizes a BART-based commonsense inference model in conjunction with an event generation model to place event-related constraints on the story generation process. We acknowledge high-level similarity in approaches between our framework and this work, but we do not include this approach in our comparisons since it is concurrent.

Storytelling research focused on improving long-range cohesion is not limited to using commonsense resources—Goldfarb-Tarrant et al. (2020) perform high-level planning via plot outline generation using principles from Aristotle’s Poetics, then use a language model in fill in details. They demonstrate strong performance on the WritingPrompts (Fan et al., 2018) dataset. While one of their models’ purpose is to determine whether to reuse or introduce a new character, they do not explicitly model inter-character relationships or character attributes during generation. We compare to this baseline in our experiments.

3 The CAST Inference Method

We now introduce our neural storytelling framework, Commonsense inference Augmented neural StoryTelling (CAST), which scaffolds the conventional text generation process by imposing constraints on the sampling process at inference time.

The conventional setup for \( k \)-sentence story generation starts with a given first sentence \( s_1 \), referred to as the prompt, and generates \( k - 1 \) subsequent sentences conditioned on it. CAST follows this convention, generating the \( i \)th sentence as follows:

1. We condition a fine-tuned language model on the story up to the current sentence \([s_1, \ldots, s_{i-1}]\), followed by a token signifying the main character of sentence \( i \) (§3.1).
2. We obtain a set of commonsense inferences for \( s_{i-1} \) (§3.2) and use them as constraints at the decoding stage of sampling a next-sentence candidate \( c \) from the language model (§3.3).
3. We obtain a set of commonsense inferences for candidate \( c \), and match commonsense inference sets between \( s_{i-1} \) and \( c \) using a matching criteria, producing a score for \( c \) (§3.2).
4. If the score is above a specified threshold, \( c \) is selected to be \( s_i \) and is appended to the gener-
An illustration of the pipeline is given in Figure 2. In practice, when generating two-character stories, we specify main characters in an alternating manner to promote turn-taking (more details in §3.1). CAST is not limited in application to the maximum story length seen during training, and can be used to generate stories of arbitrary length.

3.1 Language Model

We fine-tune GPT-2 on a story corpus to prime the model to elicit story-like generations (See Section 4.1 for details on the three story corpora we use for experiments). In order to directly compare to prior work (Guan et al., 2020), we use the small version of GPT-2, although our technique works with any neural language model.

We first pre-process the corpus to remove character names to improve generality and avoid gender bias. We replace them with character tags such as [Char_1], [Char_2] or [Char_3]. This allows our generated stories to not be limited to turn-taking between characters of different genders, in contrast to prior work (Guan et al., 2020) who use gendered tags ([MALE], [FEMALE], [NEUTRAL]) to differentiate characters.

We perform a second fine-tuning step where we append a special prefix \(T^*\) to sentence pairs. \(T\) is a character tag ([Char_1], [Char_2], etc.) representing the character that is the subject of the second sentence. Fine-tuning on this corpus allows us to specify main characters during Step 1 of the CAST inference process using the \(T^*\) tag.

This allows the model to generate a second sentence where \(T\) is the subject, but not necessarily the first word, of the sentence. For example, consider the sentence “[Char_1] has a big crush on [Char_2]”. If the next sentence in the corpus has the entity represented by [Char_2] as the subject of the event, then we concatenate \(T^*[\text{Char}_2]\) onto the first sentence during fine-tuning in order to cue the language model about turn-taking. We identify the subject entity in a sentence using a parser. We found in initial experiments that this allowed more flexibility and improved generation quality over the alternative (always requiring the main character \(T\) to be the first word of a sentence).

More details are given in Appendix A.3.

To use the language model to generate a single-character story, we always set \(T\) to the same character ([Char_1]). In a two-character story setting, we adopt character turn-taking principle: in a two-character 5-sentence story, \(T = [\text{Char}_2]\) for generating even numbered sentences and \(T = [\text{Char}_1]\) for generating odd-numbered sentences.

3.2 Generating and Matching Commonsense Inferences

To produce commonsense inferences for each sentence, we use the COMET\(_{20}\) model (Hwang et al., 2021) to infer a set of commonsense relations for each prompt sentence \(s_{i-1}\) and continuation sentence \(c\). Table 1 has the list of the subset of inferences we use and their definitions.

Once we have inferred relation sets for both sentences, we look for specific patterns between the sets of ATOMIC relations. We identify eight relation pairs that are useful for creating coherent relations between story events described in adja-
Table 1: Definitions of the selected types of ATOMIC relations that CAST uses. Those prefixed with x refer to the sentence’s subject character and those with o refer to other characters.

<table>
<thead>
<tr>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>xWant</td>
<td>as a result, PersonX wants</td>
</tr>
<tr>
<td>xIntent</td>
<td>because PersonX wanted</td>
</tr>
<tr>
<td>xNeed</td>
<td>before this event, PersonX needed</td>
</tr>
<tr>
<td>xEffect</td>
<td>as a result, PersonX will</td>
</tr>
<tr>
<td>xAttr</td>
<td>PersonX is seen as</td>
</tr>
<tr>
<td>xReact</td>
<td>as a result, PersonX feels</td>
</tr>
<tr>
<td>oReact</td>
<td>as a result, PersonY or others feels</td>
</tr>
<tr>
<td>oWant</td>
<td>as a result, PersonY or others want</td>
</tr>
<tr>
<td>oEffect</td>
<td>as a result, PersonY or others will</td>
</tr>
<tr>
<td>CausesDesire</td>
<td>This event makes PersonX wantDesires</td>
</tr>
<tr>
<td>Desires</td>
<td>PersonX desires</td>
</tr>
</tbody>
</table>

Table 2: Commonsense relation pairs that we identify as leading to meaningful sentence continuations. Table 1 provides attribute definitions.

<table>
<thead>
<tr>
<th># Characters</th>
<th>Prior Context</th>
<th>Continuation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single</td>
<td>xWant</td>
<td>xIntent</td>
</tr>
<tr>
<td></td>
<td>xReact</td>
<td>xReact</td>
</tr>
<tr>
<td></td>
<td>xEffect</td>
<td>xEffect</td>
</tr>
<tr>
<td></td>
<td>xReact</td>
<td>xAttr</td>
</tr>
<tr>
<td></td>
<td>CausesDesire</td>
<td>Desires</td>
</tr>
<tr>
<td>Multiple</td>
<td>oReact</td>
<td>xAttr</td>
</tr>
<tr>
<td></td>
<td>oWant</td>
<td>xIntent</td>
</tr>
<tr>
<td></td>
<td>oEffect</td>
<td>xEffect</td>
</tr>
</tbody>
</table>

cent sentences, five for single-character and three for two-character stories (Table 2) by analyzing semantic similarities between these relation pairs inferred by stories in ROCStories (Mostafazadeh et al., 2016). Details on the process for finding relation pairs can be found in Appendix A.4.

The relation types in the second column of Table 2 are interpreted as postconditions of a prior event because they are inferences of things that might have changed, such as an effect of the event, or a character is expected to form a new intention, or a character is expected to have a reaction. The relation types in the third column are interpreted as preconditions of any generated continuation because they are inferences about facts that needed to be established by prior events in the story, such as a character having an intention, a character having desires, or a character having a property.

CAST seeks to chain preconditions of the currently generated sentence with the post-conditions of the previous sentence as a means of checking whether readers will comprehend the continuation and perceive coherence. For example, suppose sentence $s_{i-1}$ is “[Char_1] gives [Char_2] a burger”. From this one can infer an oWant is “to thank”, indicating that [Char_2] may want to thank [Char_1]. If [Char_2] is to be the subject of the subsequent sentence, a good candidate sentence would be one from which the xIntent “to thank” can be inferred, such as “[Char_2] said thanks to [Char_1]”.

Once we have inferred relations for the previous sentence $s_{i-1}$ and a current candidate $c$, we judge the coherence of $c$ with the following procedure:

- The event in $s_{i-1}$ affects the wants of a character, which manifests as an intention of the primary character in the subsequent sentence ($x\text{Want} \rightarrow x\text{Intent}$ in a single-character story; $o\text{Want} \rightarrow x\text{Intent}$ in a multi-character story).
- An effect of the event in $s_{i-1}$ is something the primary character will do in the subsequent sentence ($x\text{Effect} \rightarrow x\text{Effect}$ in a single-character story; $o\text{Effect} \rightarrow x\text{Effect}$ in a multi-character story).
- A reaction to the event in $s_{i-1}$ should match either some property, or the reaction, of the primary character in the subsequent sentence ($x\text{React} \rightarrow x\text{Attr}$ in a single-character story and $o\text{React} \rightarrow x\text{Attr}$ in a multi-character story).
- The character’s desire in $s_{i-1}$ should be consistent in the subsequent sentence (CausesDesire $\rightarrow$ Desires in a single-character story).

To filter out “unqualified” continuations generated by the language model, we match the inference types described in Table 2 and their arguments.

In practice, we find that simple string matching does not adequately capture when two inferred relations’ arguments have slightly different phrasing (e.g., “to sleep” versus “sleeping”). We define a match as the semantic similarity between two inferences exceeding a certain threshold. To do this, we encode each relation argument into a fixed-length vector representation, and then compute the cosine similarity. We use Sentence-BERT (Reimers and Gurevych, 2019) for encoding, as it is designed for semantic similarity tasks and performs better than the traditional BERT on sentence similarity benchmarks. We use 80% semantic similarity as our threshold. In order to balance computation time and quality of the match, we require three of five and three out of three inference type pairs to match between a prompt and a candidate sentence when generating single-character and two-
character stories, respectively. Details of ablation studies on these hyperparameters can be found in Appendix A.5.

3.3 Increasing Sampling Success

CAST produces many candidates $c$—this step can be very expensive with respect to the average number of continuations needed to find a match (see Appendix A.5). In order to increase the probability of generating a continuation with a match, we use the commonsense inference set of prompt sentences as lexical constraints to control the decoding process (Peng and Sollami, 2022) when generating candidates. We first obtain the synonyms set $A$ and the antonyms set $\bar{A}$ of each commonsense inference output by COMET using WordNet (Miller, 1995). For example, if COMET infers a want for a character of “go to beach”, then $A = \{”move to beach”,”go to beach”\} and $\bar{A} = \{”leave beach”\}$. We then expand the synonyms and antonyms sets by adding conjugated forms of verbs and the plural and singular forms of nouns (De Smedt and Daelemans, 2012). For example, \{“buy dog”\} is expanded to \{“buy dogs”,”buy a dog”,”buys a dog”,”bought a dog”,...\}. Finally, we modify the conditional probability $P_{\theta}(x_i | x_{<i})$ of the language model prior to sampling as follows:

$$P_{\theta}^{\delta}(\vec{x}_i | x_{<i}) = P_{\theta}(\vec{x}_i | x_{<i}, A, \bar{A}) = \delta(\vec{x}_i, A, \bar{A}) \times P_{\theta}(\vec{x}_i | x_{<i})$$

where

$$\delta(\vec{x}_{ij}, A, \bar{A}) = \begin{cases} 1 + \mu & \text{if } x_{ij} \in A, x_{ij} \notin \bar{A} \\ 1 - \mu & \text{if } x_{ij} \notin A, x_{ij} \in \bar{A} \\ 1 & \text{otherwise} \end{cases}$$

and $\mu$ is a hyper-parameter to control the strength of the penalty. Using this altered distribution, we encourage productions of synonyms to COMET inferences and punish productions of antonyms. We only modify the probability of the top-$k$ tokens in order to maintain the fluency of generated sentences. More details are in Appendix A.6.

4 Evaluation

4.1 Datasets

We conduct experiments on three diverse storytelling datasets:

- ROCStories (ROC) (Mostafazadeh et al., 2016): contains 98,159 crowdsourced five-sentence stories involving common-sense scenarios.
- Fairy tales (FT) (Ammanabrolu et al., 2020a): contains 695 stories in the fairy tale genre scraped from user-generated stories along with their associated prompts from Reddit (r/WritingPrompts/). Average story length is 59.35 sentences.
- Writing Prompts (WP) (Fan et al., 2018): ~300K user-generated stories along with their associated prompts from Reddit (r/WritingPrompts/). Average story length is 59.35 sentences.

Table 3: Story examples generated by CAST, CAST-RL, GPT-ROC and Guan et al. (2020). The story generated by GPT-ROC-RL follows a single topic (bolded)—playing outside, and shows a good plot coherence. GPT-ROC-RL generates relatively more repetitive/boring but logically coherent narrative (in italic). Guan et al. (2020) suffers plot incoherence (underlined). More examples are given in Appendix C.

<table>
<thead>
<tr>
<th>Seed Prompt:</th>
<th>CAST:</th>
<th>GPT-ROC-RL:</th>
<th>Guan et al. (2020)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice invited Megan and the girls over for a clambake.</td>
<td>The rain was so bad that Megan couldn’t play in the rain. Alice got so sad and wanted to play in the rain with Megan. Megan got a rain coat to go outside. Alice finally went outside with Megan together.</td>
<td>Alice and Megan decided to stay inside for a week. Alice and Megan bathroom was broken. Alice and Megan got in so much trouble. Alice and Megan decided to stay inside.</td>
<td>Megan tried to be friendly with each other. Alice each could feel their favorite animal tense in their hands. Megan caught up with them. Alice wrote down the activities on a topic they would enjoy.</td>
</tr>
<tr>
<td>Alice finally wrote down the activities on a topic they would enjoy.</td>
<td>Alice finally wrote down the activities on a topic they would enjoy.</td>
<td>Alice finally wrote down the activities on a topic they would enjoy.</td>
<td>Alice finally wrote down the activities on a topic they would enjoy.</td>
</tr>
</tbody>
</table>

4.2 Baselines

We evaluate CAST against three strong baselines.

- Guan et al. (2020): fine-tunes GPT-2-Small on ATOMIC and ROCStories using a multi-objective training procedure. This baseline serves to demonstrate whether a neural language model can get everything it needs directly from a static commonsense dataset without inference and constraints. We retrain the model on the pre-processed version of the ROCStories corpus that does not contain gender tags ($§3.1$) as well as with the additional fine-tuning step for character-conditioned generation ($subsection A.3$), in order to be directly comparable to CAST in a two-character setting. Further training details can be found in Appendix A.2.
- Goldfarb-Tarrant et al. (2020): a plot-generation language model along with an ensemble of
rescoring models on Writing Prompts dataset. The system trained BART (Lewis et al., 2020) to learn to generate plots on the given prompt and then transform them into a story. We compare CAST to Goldfarb-Tarrant et al. (2020)—one of the strongest story generators on Writing Prompts dataset—to show that CAST can be generalized to other datasets.

- C2PO (Ammanabrolu et al., 2021): uses COMET to generate successor and predecessor events for a single character, performing a bi-directional search from a given start event and an end event. It uses COMET to generate successor and predecessor events directly instead of constraining a more conventional language model as is the case with CAST. As such it is a strong baseline, especially considering it uses an extra piece of input—the story ending—that can influence perceptions of coherence. For fair comparison, we follow Ammanabrolu et al. (2021) to extract high-level plots from fairy tale stories and then use the first plot as prompt and the second plot as goal for guiding C2PO.

We use the provided checkpoints of the latter two models. We thus only evaluate these systems on single-character stories, since C2PO is single-character story generator and Goldfarb-Tarrant et al. (2020) is not trained to generate stories with the number of characters chosen by humans.

### 4.3 Metrics

Given the well-established unreliability of automated metrics for creative text generation, human-participant evaluation is generally held as the gold-standard evaluation technique (Celikyilmaz et al., 2020; Caglayan et al., 2020; van der Lee et al., 2021). Consequently, we also use human-participant evaluation. We provide human participants with a pair of stories from two systems, and ask them the following questions modified from Purdy et al. (2018):

- Which story better follows a single topic: for insight into perceptions of global coherence for the entire story.
- Which story’s sentences make more sense given sentences before and after them: to evaluate local causality and commonsense reasoning in the story.
- Which story is more enjoyable: indicates story value and interestingness.
- Which story uses more fluent language: indicates story readability and grammaticality.

Similar questions have been used in evaluations of other story generation systems (cf. Tambwekar et al., 2019; Ammanabrolu et al., 2020b, 2021; Castricato et al., 2021). Each pairwise comparison is seen by at least 5 participants.

We conduct our studies using the Cloud Research crowdsourcing platform to interface with Amazon Mechanical Turk (Litman et al., 2017). Only those who pass a screening question are qualified for the study. Participants must also explain their preferences for each comparison with more than 50 characters of free text. We manually verify screening question responses to qualify participants and disregard data for those who fail the screening. All crowdsourcing studies we conducted were approved by our institution’s Institutional Review Board (IRB). We recruited 86 participants from the and bi-grams, which does not necessarily entail plot-level repetition; the same entities can make appearances in different events in different ways.

<table>
<thead>
<tr>
<th>Models</th>
<th>Data set</th>
<th>Num set</th>
<th>Logical Sense Win% Lose%Tie%</th>
<th>Single Topic Win% Lose%Tie%</th>
<th>Enjoyable Win% Lose%Tie%</th>
<th>Fluency Win% Lose%Tie%</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAST vs Guan et al.</td>
<td>ROC</td>
<td>1</td>
<td>92.0** 4.0 7.0</td>
<td>60.0** 7.0 3.8</td>
<td>87.0** 4.0 9.0</td>
<td>87.0** 4.0 9.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>85.8** 6.6 7.5</td>
<td>62.9** 8.6 5.8</td>
<td>81.1** 12.3 6.6</td>
<td>83.0** 9.4 7.5</td>
</tr>
<tr>
<td>CAST vs Goldfarb-Tarrant et al</td>
<td>WP</td>
<td>1</td>
<td>64.2* 32.1 3.8</td>
<td>64.2** 28.3 7.5</td>
<td>62.3** 26.4 11.3</td>
<td>52.8 34.0 13.2</td>
</tr>
<tr>
<td>CAST vs C2PO</td>
<td>FT</td>
<td>1</td>
<td>81.5** 9.3 9.3</td>
<td>63.6** 23.6 12.7</td>
<td>81.8** 10.9 7.3</td>
<td>85.5 5.5 9.1</td>
</tr>
</tbody>
</table>

Table 4: Human-participant evaluation results for experiments 1 and 2, showing the percentage of participants who preferred the first system, second system, or thought the systems were equal. 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 Wilcoxon sign test on win-lose pairs. See results about majority votes and agreement in Table 10.
United States, paying $11.7 per hour on average. Only those with HIT approval rate above 90% and have over 1000 HITs approved were selected. Average inter-annotator agreement, measured by Fleiss’ kappa (Fleiss, 1971), is > 0.2 (fair); a more detailed breakdown by experiment can be found in Table 10. Further details are provided in Appendix B.

We randomly select a subset of first-sentences from the test sets of each dataset—20 each of 1-character and 2-character prompts from ROCStories, 10 prompts from WP, and 10 from FT. We use these sentences to generate a story continuation of 4 sentences from each system. We recruited 86 participants on a crowdsourcing platform. Each participant answered the four pairwise comparison questions (§4.3) on a randomly selected subset of 5 story pairs, comprised of one story from CAST and one from one of the baselines.

4.4 Results

The results are shown in Table 4 (top) where we detail the percentage of times human participants choose the story from one system over another for each dimension in the questionnaire. We indicate when results are significant at $p < 0.05$ and $p < 0.01$ confidence levels. Generally, participants strongly preferred stories generated by CAST to those generated by alternatives.

Compared with Guan et al. (2020), CAST is able to find a commonsense inference link to develop the stories on ROCStories prompts, which makes it much more coherent and stay on one single topic. Human participants state in their response that stories generated by CAST have better commonsense flow and make more sense. Stories generated by CAST is also more enjoyable and fluent because of its high coherence.

Since COMET is trained on ROCStories, we also seek whether CAST works on other datasets. We compared CAST to Goldfarb-Tarrant et al. (2020) on Writing Prompts, which contain longer and more complicated stories. CAST with language model fine-tuned on Writing Prompts outperforms Goldfarb-Tarrant et al. (2020) in “Logical Sense”, “Single Topic” and “Enjoyable” dimensions. On the topic of fluency, CAST is preferred but the result is not statistically significant when ties are considered. Human participants stated that they found stories generated by CAST is much easier to follow and they are built on a single topic. Because Goldfarb-Tarrant et al. (2020) applies BART (Lewis et al., 2020) to generate plots, which cannot ensure commonsense like CAST.

C2PO is also built on COMET to conduct a bi-directional search from a given start event and a given end event, which makes it as a strong baseline to compare. We follow Ammanabrolu et al. (2021) to extract high-level plots from fairy tale stories as prompts and goals for evaluation. CAST outperforms C2PO on all dimensions, because we apply a harder commonsense constraints on continuation generation than C2PO, which produce a more coherent and on-topic story. We anecdotally observe that CAST generates more diverse stories than C2PO because of templated and limited range of COMET, which we only use for filtering whereas C2PO uses it for sentence generation.

We conclude that CAST is able to produce a much more coherent, on-topic, enjoyable and fluent story than strong baselines. It also has the advantage over Goldfarb-Tarrant et al. (2020) and C2PO for choosing characters in the story continuations, which makes CAST able to produce single- or two-character stories.

5 Conclusions

Neural language models generate content based on the likelihood of tokens given a historical context. Human readers, on the other hand, use complex inferential processes to connect the relationships between events. This mismatch between generative models and reader comprehension is one of the reasons why stories generated by neural language models lose coherence over time.

Our CAST system is a straightforward approach to enforce the constraint that a language model only generate continuations that cognitive psychology tells us will be more comprehensible. The CAST method provides hard constraints to neural language model generation that results in greater story coherence, a result that holds in multiple storytelling domains. We find that perceived story enjoyability and fluency are tied to making logical sense, tracking character goals, and staying on topic; our system excels in all four of these areas.

We seed prompts and goal sentences for C2PO. For 2-character stories, we use the interleaving story generation method described in §3.1.

We observe that these four dimensions are highly, positively correlated using Spearman’s Rank Order Correlation (See Appendix B.3).

We only provide prompts to CAST.
6 Acknowledgements

This work was done while SW was at the Georgia Institute of Technology.

7 Limitations

The primary data source of our paper is the ROC-Stories dataset. ROCStories consists of many event-centric narratives which, while often used in story generation research, is still not representative of complex, realistic narratives. This may give COMET\textsuperscript{20} (Hwang et al., 2021) an advantage in making inferences that are used for filtering.

COMET\textsuperscript{20} requires a clearly identifiable actor in each sentence in order to make commonsense inferences for that actors. Thus our language model—by virtue of fine-tuning on ROCStories—produces sentences (events) that have an identifiable character performing an action. Stories can have more complex expository text. Narratologists—those that study narratives—often distinguish between events—text that implies a change to the world and thus drive the story forward—and exposition—text that describes elements of the story world without changing it.

The performance of CAST is tied to the inference abilities of COMET\textsuperscript{20}. As such, the types of errors that COMET is prone to are also the types of errors that our system is prone to. We invite readers to review the discussion in Hwang et al. (2021) for more detailed analysis of commonsense inference errors. As more advanced commonsense inference models develop, CAST-like approaches will benefit from the improved state-of-the-art. CAST can easily switch to new generative language models or new commonsense inference model.

Restricted by the filter—COMET\textsuperscript{20}—our system works mostly for narratives with event-centric commonsense knowledge. Even though we processed the datasets (Appendix A.1) to decrease the gender biases, there is no guarantee to entirely eliminate these biases.

CAST produces stories by chaining sentence-level COMET inferences to track important implicit elements of the story between adjacent sentences. We make a Markovian assumption by only comparing the currently generated event to the most recent event. Stories are arguably non-Markovian and can have complex, interleaving chains of inference; despite the assumption, we find in practice that it enforces global coherence quite successfully (see Single Topic metric in Table 4). Future work may relax this constraint by keeping track of wants/needs/etc from previous sentences against which to match. One would need to solve the problem of deciding when wants/needs/etc should expire because they are no longer applicable.
References


and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.


A Implementation Details

A.1 Data

We use the preprocessed ROCStories corpus of 5-sentence stories (588,966 stories) and joint ATOMIC and ConceptNet dataset converted to template sentences (1,174,267 train/66,856 dev/73,083 test), both provided by Guan et al.\textsuperscript{7} We shuffle and split the ROCStories dataset into 80\% (78,528) train and 20\% (19,633) test sets.

Following Guan et al.\textsuperscript{(2020)}, character names in the ROCStories corpus are replaced with [MALE] or [FEMALE] tags. In order to remove gender bias, we replace [MALE], [FEMALE], or [NEUTRAL] tags with [Char\_1], [Char\_2], or [Char\_3] tags. This prevents skewed predictions due to the presence of certain names in a small dataset such as ROCStories and also allows us to focus on 2-character stories without having to perform NER on generated sentences to remove extraneously generated names outside of the two main characters. It also allows a direct comparison to prior work. After a story is generated, we replace the character tags with user-inputted names, assuming the subject and object of the first sentence are the subsequently-generated tags.

A.2 Models

Following Guan et al., we use the small version of GPT-2 with 124M parameters as the base for our fine-tuned models. When fine-tuning GPT-2 on either ROCStories and the commonsense knowledge resources (done separately), we train with a learning rate of 0.00005, and using the Adam optimizer with gradient clipping at a max norm of 1. CAST and Guan et al.\textsuperscript{(2020)} were trained on single GeForce RTX 2080 GPUs in Pytorch using the Huggingface Transformers library.\textsuperscript{8} We replicate the multi-task baseline of Guan et al. in Tensorflow using their provided code.\textsuperscript{9} We train with early stopping on the dev set (80\% train, 10\% dev, 10\% test split) loss with a patience of 10 epochs. Both models converge within 1-2 epochs. All other training details are kept the same. We use top-p sampling (Holtzman et al., 2019) with a value of 0.9, a temperature of 1, and a max length of 20 tokens per sentence to sample from CAST and Guan et al.\textsuperscript{(2020)}.

For Goldfarb-Tarrant et al.\textsuperscript{(2020)}, we use BART model using the code and parameters published on the paper’s public repository.\textsuperscript{10}

We replicate the C2PO model by Ammanabrolu et al.\textsuperscript{(2021)} using the code published on the paper’s public repository.\textsuperscript{11} All the encoder and model checkpoints are provided by the author.

A.3 Character Conditioned Generation

To enforce the telling of a two-character narrative in an interleaving fashion wherein characters take turns being the subject of each sentence. We fine-tune the language model by formulating the input as \( s^T \) \( [s_1, \ldots, s_{i-1}] \), where \( T \) is the tag denoting the character who is to take a turn, which is determined by Part-Of-Speech Tagger\textsuperscript{12}. For example, for the story, “[Char\_1] was upset with [Char\_2]. Because of this, [Char\_2] apologized.”, the prompt is formulated as “* [Char\_2] * [Char\_1] was upset with [Char\_2].” The language model is fine-tuned by back-propagating the loss calculated on the sentence “Because of this, [Char\_2] apologized.” At test-time, we generate second sentence candidates until one contains a reference to [Char\_2].

A.4 Commonsense Matching Criteria

We randomly selected 500 stories from ROCStories\textsuperscript{(Mostafazadeh et al., 2016)}. We then use COMET to produce commonsense inference sets for these stories of all the 34 relations in ATOMIC with beam size of 10. Hence, for all the 500 \( \times \) 5 sentences, we obtain 10 commonsense inference for each type (34 types). For each sentence, we consider the commonsense relation sets of current sentence and its next sentence as a pair. So we have \( 32 \times 31 \) pairs for each pair of the adjacent sentences. Then we adopt Sentence-BERT (Reimers and Gurevych, 2019) to encode all these inference and calculate the max cosine similarity of each commonsense inference pair for each adjacent sentence pair. Inference pairs with over 80\% semantic similarities are used as hard constraints via a form of chaining that allows us to filter a set of potential sentence generations to find one that adequately matches the expected inferences.

\textsuperscript{7}https://cloud.tsinghua.edu.cn/d/670f7787b6554f308226/
\textsuperscript{8}https://huggingface.co/transformers/
\textsuperscript{9}https://github.com/thu-coai/CommonsenseStoryGen
Table 5: Ablation study result for semantic similarity. We run CAST without controlling decoding to generate 20 2-char stories in 5 seeds. # of sentence candidates denotes the average number of sentences candidates generated before finding a matching inference type pair. Success rate is the percentage of finding a match within the 50-candidate limit. A lower Self-BLEU score implies more diversity of the document (Zhu et al., 2018) (see §4.3).

<table>
<thead>
<tr>
<th>Semantic Similarity</th>
<th># of Sentence Candidates ↓</th>
<th>Success Rate ↑</th>
<th>Self ↓ BLEU-2</th>
<th>Self ↓ BLEU-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8</td>
<td>11.25</td>
<td>98.75%</td>
<td>.1718</td>
<td>.0892</td>
</tr>
<tr>
<td>0.85</td>
<td>17.25</td>
<td>97.25%</td>
<td>.1978</td>
<td>.1140</td>
</tr>
<tr>
<td>0.9</td>
<td>18.05</td>
<td>91.25%</td>
<td>.2011</td>
<td>.1216</td>
</tr>
</tbody>
</table>

Table 6: Story examples generated by CAST with different semantic similarity thresholds. Stories generated at 80% similarity maintains more diversity.

Table 7: Ablation study result for required matching inference type pairs in single-character stories. We run CAST without controlling decoding to generate 30 single-character stories in 5 seeds. # of sentence candidates denotes the average number of sentences candidates generated before finding a matching inference type pair. Success rate is the percentage of finding a match within the 50-candidate limit. The number of sentence candidates could Failure to find a match within the candidates limit (50) will relax the matching constraints to one pair. Hence, the average number of sentences candidates might be over the candidate limit. A lower Self-BLEU score implies more diversity of the document (Zhu et al., 2018) (see §4.3).

<table>
<thead>
<tr>
<th># of Matching</th>
<th># of Sentence Candidates ↓</th>
<th>Success Rate ↑</th>
<th>Self ↓ BLEU-2</th>
<th>Self ↓ BLEU-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>9.78</td>
<td>98.67%</td>
<td>.1559</td>
<td>.0860</td>
</tr>
<tr>
<td>4</td>
<td>18.95</td>
<td>86.83%</td>
<td>.1832</td>
<td>.1282</td>
</tr>
<tr>
<td>5</td>
<td>56.39</td>
<td>40.17%</td>
<td>.1870</td>
<td>.1356</td>
</tr>
</tbody>
</table>

A.5 Ablation Study of Commonsense Inferences Matching

We use 80% semantic similarity as our lower-bound. Empirically, we find this value best considers the inferences listed in Section 3.2 as matches, but excludes less-related inferences. Table 6 shows how the threshold affects success rate—the percentage of queries that find a match within 50 generated candidates—and the diversity of results as measured by self-BLEU score (described in §4.3). Each system was conditioned on the same 20 2-character prompts from ROCStories with 5 different random seeds, requiring two of three inference

https://github.com/PlusLabNLP/story-gen-BART
https://github.com/rajammanabrolu/C2PO
https://nlp.stanford.edu/software/tagger.shtml
type pairs to match to qualify as a match. Failure to find a match within the candidates limit (50) will relax the matching constraints to two pairs. Hence, the average number of sentences candidates might be over the candidate limit. As observed in 5, increasing the semantic similarity threshold decreases the success rate in obtaining a matching candidate within the sentence limit, and it results in more repetitive sentences (see Table 6).

In order to balance computation time and quality of the match, we only require three of five inference type pairs to match between a seed and a candidate sentence when generating single-character stories. When requiring five matches when generating single-char story, CAST only finds a “qualified” sentence 40% of the time within 50 attempts (see Table 5 (bottom), computed at 0.8 semantic similarity). In practice (see examples in Table 8), we find requiring three pairs results in higher quality sentences than if we only require one or two out of three pairs to match, but is significantly more efficient than four or five out of five.

A.6 Decoding Process Ablation Study

<table>
<thead>
<tr>
<th>Decoding</th>
<th># of Sentence Candidates</th>
<th>Success Rate</th>
<th>Self ↓</th>
<th>Self ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>3.21</td>
<td>99.50%</td>
<td>.1709</td>
<td>.0913</td>
</tr>
<tr>
<td>False</td>
<td>11.25</td>
<td>98.75%</td>
<td>.1718</td>
<td>.0892</td>
</tr>
</tbody>
</table>

Table 9: Ablation study result for required controlling decoding stage. We run CAST with or without controlling decoding to generate 20 multiple-character stories in 5 seeds. # of sentence candidates denotes the average number of sentences candidates generated before finding a matching inference type pair. Success rate is the percentage of finding a match within the 50-candidate limit. A lower Self-BLEU score implies more diversity of the document (Zhu et al., 2018).

In order to increase the probability of finding a match in Section 3.2, inspired by Peng and Sollami (2022), we use commonsense inferences of prompt sentences as lexical constraints to control the generation decoding process. We run a ablation test to validate this component of CAST. Table 9 shows that after applying commonsense inferences of prompt sentences as lexical constraints to control the generation decoding process, CAST successfully find a match in the average of 3 candidates. At the same time, self-BLEU score did not show any statistical difference. Hence, we adopt decoding technique in CAST.

A.7 CAST

When producing commonsense inferences from COMET, we use “beam-5” setting to generate 5 inferences for each inference type, which results in a higher percent of matched inferences in our preliminary experiments. We also qualitatively observe that matching on a larger set of inferences (as shown in the demo 13) more often results in at least one or a few high-quality inferences, due to COMET having some error.

As mentioned in the body of the text, we use a semantic similarity threshold of 80% and require 3 of 5 inferences to match when generating single-character stories. Runtime is feasible due to matching on three out of five inference filters and using the 5-beam COMET output. However, in some rare cases, no matching next-sentence candidate can be found. If no qualified sentence is found after 50 generated candidates, in order to avoid potentially infinite search we loosen the filtering strength to match only one pair of inferences. We also report the majority vote of experiments in Table 10.

13 https://mosaickg.apps.allenai.org/comet_atomic
Table 10: Human-participant evaluation results for experiments 1 and 2, showing the percentage of participants who preferred the first system, second system, or thought the systems were equal. 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 Wilcoxon sign test on win-lose pairs. † indicates $\kappa > 0.2$ or fair agreement. ‡ indicates $\kappa > 0.4$ or moderate agreement.
B Evaluation

B.1 Evaluated Stories Generation

In order to compare with Guan et al. (2020), we randomly select a subset of first sentences of ROC-Stories as prompts to seed CAST and Guan et al. (2020), generating 5-sentence stories from each model. We considered two cases—(1) single-character and (2) two-character stories. In order to generate two-character stories, we seed the story history and continuation’s subject to GPT-2. More details can be found in Appendix A.3. We use a subset of prompts given by Writing Prompts to seed our system and Goldfarb-Tarrant et al. (2020). We also keep 5 sentences to evaluate the models. Since C2PO is controllable story generation model trained on fairy tale stories, we seed the first sentence in the fairy tale stories as prompt and the 5th sentence in the story as goal to C2PO for generating stories. For CAST model, we only seed the first sentences of the fairy tale stories. Examples can be found in Appendix C.

B.2 Human Study Setup

We show human participants instructions (Fig. 3) and then they are required to pass screen questions (Fig. 4). They then answer which story best met the criteria, as shown in Fig. 5.

B.3 Correlation between answers

We compute the Spearman rank correlations between the workers’ different answers for the story pairs they are responsible for rating. We ignore workers who did not complete all of the questions in our computations. Our results are displayed in
<table>
<thead>
<tr>
<th>Study</th>
<th>1 &amp; 2</th>
<th>1 &amp; 3</th>
<th>1 &amp; 4</th>
<th>2 &amp; 3</th>
<th>2 &amp; 4</th>
<th>3 &amp; 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAST vs. C2PO</td>
<td>0.39*</td>
<td>0.34</td>
<td>0.34</td>
<td>0.47*</td>
<td>0.77*</td>
<td>0.39*</td>
</tr>
<tr>
<td>CAST vs. Goldfarb-Tarrant et al.(2020)</td>
<td>0.58*</td>
<td>0.44*</td>
<td>0.38*</td>
<td>0.56*</td>
<td>0.35</td>
<td>0.47*</td>
</tr>
<tr>
<td>CAST-ROC-1Char vs. Guan et al.(2020)</td>
<td>0.51*</td>
<td>0.28*</td>
<td>0.43*</td>
<td>0.67*</td>
<td>0.65*</td>
<td>0.67*</td>
</tr>
<tr>
<td>CAST-ROC-2Char vs. Guan et al.(2020)</td>
<td>0.53*</td>
<td>0.49*</td>
<td>0.39*</td>
<td>0.91*</td>
<td>0.79*</td>
<td>0.79*</td>
</tr>
<tr>
<td>CAST-RL-1Char vs. Guan et al.(2020)</td>
<td>0.78*</td>
<td>0.65*</td>
<td>0.47*</td>
<td>0.67*</td>
<td>0.54*</td>
<td>0.45*</td>
</tr>
<tr>
<td>CAST-RL-2Char vs. Guan et al.(2020)</td>
<td>0.51*</td>
<td>0.45*</td>
<td>0.44*</td>
<td>0.65*</td>
<td>0.65*</td>
<td>0.55</td>
</tr>
<tr>
<td>CAST-RL-1Char vs. CAST-ROC-1Char</td>
<td>0.83*</td>
<td>0.28*</td>
<td>0.55*</td>
<td>0.32*</td>
<td>0.64*</td>
<td>0.31*</td>
</tr>
<tr>
<td>CAST-RL-2Char vs. CAST-ROC-2Char</td>
<td>0.14</td>
<td>-0.05</td>
<td>0.25</td>
<td>0.27</td>
<td>0.39*</td>
<td>0.30*</td>
</tr>
</tbody>
</table>

Table 11: Spearman correlation results from all our human subject studies. The four numbers 1, 2, 3, and 4 corresponds to “Follow a single topic”, “Logical Sense”, “Enjoyable”, and “Fluency” metrics respectively. Spearman correlations with $p \leq 0.01$ are marked with an *. 
Figure 4: Screening questions used to qualify participants for the main study. Correct answers for these questions are [2,1,any,2], where 1 indicates the first story is the correct answer, 2 indicates the second one is the correct answer and “any” indicates no correct answers, any answer can pass the screen question.

Figure 5: User interface for the main human evaluation task. Each participant completes 5 such tasks in a single HIT.
C Additional Story Outputs

Additional 5-sentence story examples are as follows,

Seed Prompt:
Bob and Alice went hiking together.

CAST:
Alice was excited because she planned a picnic.
Bob was excited because he was staying up late at night.
Alice felt much better about her camping trip.
Bob spent the night playing in the woods.

CAST-RL:
Alice decided to take a group photo.
Bob shared the picture with Alice.
Alice came over and smiled at Bob.
Bob was so happy that he took the picture.

Guan et al. (2020):
Alice thought that the hike was cool.
Bob was pretty cool.
Alice reminded him of the great fall, especially in the forest.
Bob was cute, but Alice was lost.

Seed Prompt:
Bob asked Alice to come over for dinner.

CAST:
Alice wanted to go to a party.
Bob agreed to go with her.
Alice and Bob went to the concert.
Bob bought a bottle of wine, and they enjoyed their time.

CAST-RL:
Alice said yes, so he brought her food.
Bob had dinner with Alice that night.
Alice said yes to the dinner.
Bob was happy that he invited her to dinner with him.

Guan et al. (2020):
Alice drove and came over Bob’s house in time.
Bob decided to let Alice choose the foods.
Alice ate some dog food and then she met her husband.

Bob offered Alice a side of anything he had for dinner.

Seed Prompt:
Bob took Alice fishing in the summer.

CAST:
Alice waited at the lake for hours.
Bob got another boat and brought it back home.
Alice and Bob fished the next day.
Bob enjoyed the fishing and had a great time.

CAST-RL:
Alice caught ten fish.
Bob was pleased that he caught ten fish.
Alice was very pleased to have a good fish.
Bob is glad that he caught ten fish.

Guan et al. (2020):
Alice got seasick.
Bob asked Alice to help him.
Alice went to Bob’s house to tow Bob’s boat.
Bob got a surfboard for a big fishing trip.

Seed Prompt:
Bob was engaged to Alice.

CAST:
Alice wanted a diamond ring.
Bob thought that ring was the perfect diamond.
Alice loved the diamond ring and decided to buy it.
Bob went to the store and bought the ring.

CAST-RL:
Alice was going on a date with Bob.
Bob agreed to the date and married her.
Alice now has two kids and needs the baby.
Bob is very happy he finally realized she was unhappy.

Guan et al. (2020):
Alice just got a new husband.
Bob wondered what to expect.
Alice bought the nice wedding ring from a daycare.
Bob wore it to the wedding.

Seed Prompt:
Bob took Alice to school today.
CAST:
Alice was a freshman.
Bob took her to the library.
Alice was excited to take the long bus ride to the library.
Bob continued the long walk through the library.

CAST-RL:
Alice had been waiting for hours at school.
Bob was relieved to see her take the bus to school.
Alice was excited to see the bus arrive.
Bob was proud of her for taking the bus to school.

Guan et al. (2020):
Alice was nervous because Bob didn’t know how to tap dance.
Bob was excited for his first attempt at tap dancing.
Alice was impressed and wanted to learn.
Bob started his first dance at the dance school.

Additional 10-sentence story examples are as follows:

Seed Prompt:
Bob went on her very first date with Alice.

CAST:
Alice was very excited about having a date with Bob.
Bob was excited and eager.
Alice informed Bob she was very excited for him.
Bob was very appreciative.
Alice said it was a perfect time.
Bob and Alice had a wonderful first date.
Alice was surprised she was invited to his birthday.
Bob was thrilled.
Alice was happy to have a new boyfriend.

Guan et al. (2020):
Alice and Bob didn’t realize that Bob loved movies.
Bob’s friend, Alice, saw the movie with Alice.
Alice and Bob were good friends.
Bob’s friend, Alice, screamed for Alice.
Alice was their eyes met her eyes.
Bob and Alice’s grandparents were never match.
Alice because they had ever thought, they thought the end the words said, she passed with Alice.
Bob and they agreed that they went out the eyes.
Alice felt they got puppies and they do the thought she was their eyes Alice had.

Seed Prompt:
Bob broke up with Alice.

CAST:
Alice was very hurt and very angry.
Bob knew he would have to live with his pain.
Alice said she could give him a relief.
Bob felt much better after that.
Alice felt even better in peace.
Bob is now happy with himself.
Alice felt much better.
Bob will make himself a happy man.
Alice feels much better after having peace.

Guan et al. (2020):
Alice didn’t want to get it.
Bob thought that he was too thin.
Alice said he could give back inside a horse.
Bob could bring along for a ride and to return to the river.
Alice had to drive for three days and she realized she was cheating.
Bob and Alice got into a fight.
Alice ran away with Alice and grounded.
Bob and sent Alice.
Alice and a girl kicked each other.

Seed Prompt:
Bob went to see Alice today.

CAST:
Alice told him about the movie she loved to watch.
Bob grabbed a movie poster from the library.
Alice shared her own reaction to the movie.
Bob watched the movie that night.
Alice felt very satisfied after the movie was over.
Bob went home happy that he watched the movie.
Alice has been a huge fan of the movie ever since.
Bob looks forward to a nice vacation at the movies.
Alice is glad that she shared her own comments to the movie.

Guan et al. (2020):
Alice was nice and gregarious.
Bob had a great time.
Alice had some great food and wine.
Bob couldn’t wait to see Alice again.
Alice let him take a bath.
Bob felt smarter and had a good time again. Alice also got his great seeing Alice tattoo. Bob had a good time. Alice tattoooy from his favorite j tomorrow too.

---

**Seed Prompt:**

Bob went over to play at Alice’s house.

---

**CAST:**

Alice liked to watch movies. Bob was always there watching movies. Alice saw a movie she wanted to see. Bob was very glad to buy that movie. Alice was so happy with the movie he bought. Bob was excited to be watching the movie with Alice. Alice could not wait to see the movie with Bob. Bob and Alice had so many fun. Alice laughed and thanked Bob for the great movie.

---

**Guan et al. (2020):**

Alice was a baby dolls. Bob told Alice he could save a doll. Alice went to the video game section to change the game. Bob was amazed to see who did the laundry to the last minute. Alice waved and proposed the doll girls. Bob wished her that she was getting a doll. Alice helped him with the $40 and she gave her $100, but by buttering buttering a doll. Bob’s doll broke. Alice had an ruined doll and they would buy her.