Augmented Neural Story Generation with Commonsense Inference

Anonymous ACL submission

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 neural StoryTelling (CAST), a framework for introducing commonsense reasoning into the generation process while modeling the interaction between multiple characters. We find that our CAST method produces significantly more coherent and on-topic two-character stories, outperforming baselines in dimensions including plot plausibility and staying on topic. We also show how the CAST method can be used to further train language models that generate more coherent stories and reduce computation cost.

1 Introduction

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

A common approach to story generation is to use neural language models (Roemmele, 2016; Khalifa et al., 2017; Clark et al., 2018; Martin et al., 2018). These techniques have improved with the adoption of Transformer-based models, such as GPT-2 (Radford et al., 2019). While GPT-2 and similar neural language models are considered highly fluent from a grammatical standpoint, they 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. One reason for these phenomena is that language models generate continuations by sampling from a learned distribution $P_\theta(tok_n|tok_{<n})$. Human readers, however, do not perceive the coherence of a narrative as a function of the likelihood of seeing particular words based on the occurrence of previous words.

Previous attempts to enhance the coherence of generated stories 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 achieve coherence through adherence to a manually given high-level plan. The high level plan can be generated (Yao et al., 2019; Am-

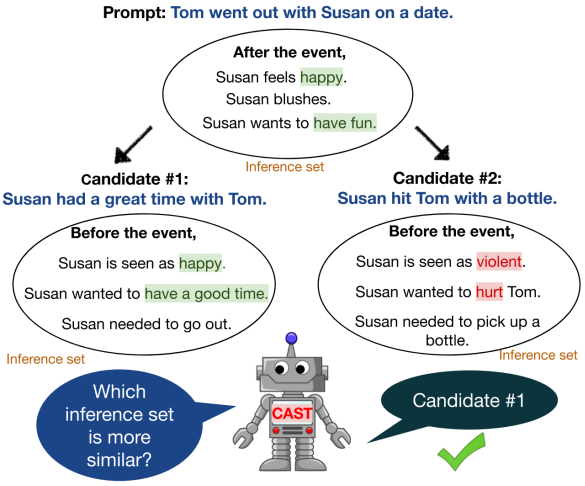

Figure 1: Overview of the CAST system. 1. A text prompt starts the story generation process. 2. The system infers facts about the characters' intents. 3. A language model generates candidate continuations. 4. Inferences about the candidates are matched against the previous inferences and the best candidate is added to the story.

manabrolu et al., 2020), which only elevates the challenges of maintaining coherence to a higher level of abstraction. Neural language models can also be fine-tuned on extraneous 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.

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. 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 character intentions and action consequences will generate more coherent stories.

However, stories don't always explicitly declare the goals and motivations of characters; sentences describing character actions are not explicitly annotated with the characters' motivations and goals. Readers must infer the characters' goals and the relationship between their actions and those goals. The ability to use basic knowledge about goals and about what is happening in the world falls within the study of *commonsense inference*. ATOMIC (Sap et al., 2019) is a commonsense knowledge base that contains logical relationships concerning mental states, attributes, and events. COMET (Bosselut et al., 2019) is a transformer-based generative model trained on triples from ATOMIC and infers relations about sentences broadly divided into four categories: (1) Causes of a person's actions (preconditions of the event), (2) Attributes of a person, (3) Effects of actions on a person (postconditions of the event), and (4) Effects of actions on others (postconditions of the event). We propose to infer character intentions and effects of actions using COMET to inform the generation of subsequent sentences by a neural language model.

To address the challenge of maintaining coherence in language-model-based story generation, we propose a novel two-character story generation method, ***C**ommonsense inference **A**ugmented neural **S**tory**T**elling* (CAST), that infers the causal relations between events as well as the intents and motivations of characters in the story context so far in order to generate story continuations that are more coherent to readers. CAST uses these inferred causal relations and character intentions to make more informed choices about potential story continuations generated by a neural language model (GPT-2). 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.

To evaluate the efficacy of our proposed method, we conduct a series of human-participant experiments specifically measuring perceptions of logical coherence of CAST against a number of fine-tuned variants of GPT-2. Results indicate that the CAST method significantly increases the perception of generated stories over baselines.

2 Related Work

Neural networks—recurrent and transformer-based—have been used to produce stories (Roemmele, 2016; Khalifa et al., 2017; Martin et al., 2018; Clark et al., 2018). In these systems a neural language model learns to approximate the distribution $P_\theta(tok_n|tok_{<n})$. Stories are generated by providing a context sequence and sampling from the distribution. When the language model is trained on a corpus of stories, the generated text tends to also be a story. Sometimes generation is done hierarchically (Yao et al., 2019; Ammanabrolu et al., 2020). However, coherence is not guaranteed; 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. Other artifacts of the sampling process include new characters being arbitrarily introduced at any time, characters being forgotten, and repetitions.

To control the generation, sometimes a high-level plot outline is given and a language model is conditioned or otherwise guided by the high-level plot outline (Fan et al., 2018; Peng et al., 2018; Rashkin et al., 2020; Brahman and Chaturvedi, 2020). These high-level guidance specifications

turn the story generation problem into a supervised learning problem. We do not consider these approaches further in this paper because we do not assume the existence of any guidance specification.

The story generation system by Guan *et al.* (2020) first fine-tunes GPT-2 on the ATOMIC dataset and then fine-tunes a second time on the ROCStories corpus (Mostafazadeh et al., 2016) to induce GPT-2 to generate short stories. The authors use multi-task learning during the second fine-tuning stage with an auxiliary objective to distinguish true and engineered false stories. Even with such specialized learning, the resulting model still fails to avoid logical errors, repeats pieces of narratives, and introduces unrelated entities. This demonstrates the need for a stronger inductive bias on how commonsense knowledge is used in story generation. Brahman and Chaturvedi (2020) generate stories that follow a given emotional arc for a protagonist, using COMET to infer the protagonist's emotions. 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. However, C2PO generates plots made up of highly constrained, templated text.

3 The CAST Method

The conventional set up for neural k-sentence story generation is: given the first sentence s_1 of a story—a prompt—generate the subsequent $k-1$ sentences, $s_2, s_3, ..., s_k$. The ***C**ommonsense inference **A**ugmented neural **S**tory**T**elling* (CAST) method is as follows. To generate sentence s_i with $2 \leq i \leq k$:

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 subject of sentence i.

2. We sample a sentence candidate c from the language model and obtain a set of commonsense inferences.

3. We match commonsense inference sets between s_{i-1} and c using a matching criteria grounded in theory, and produce a score for c.

4. If the score is above a user-specified threshold, c is selected to be s_i and is appended to the generation history. Otherwise, steps 2 through 4 are repeated until a viable candidate is found.

5. Repeat steps 1 through 4 until $k-1$ sentences have been generated.

An overview of the pipeline is given in Figure 2. The system is simple but effective—CAST reasons about whether there are sufficient relations between the candidate sentence and the previous sentence to support reader comprehension. Unlike models trained or fine-tuned on commonsense knowledge, the filtering technique provides a hard constraint that is likely to persist throughout a generated story of arbitrary length without degeneration, and it is not limited in application to the maximum story length seen during training.

3.1 Language Model

We fine-tune GPT-2 on a pre-processed version of the ROCStories corpus (Mostafazadeh et al., 2016) to encourage story generation. Following Guan et al. (2020), we pre-process the corpus to remove character names, replacing them with `[MALE]` or `[FEMALE]`. During pre-processing, we annotate characters with `*[`T`]*` where T is a character tag when that character appears in the next sentence in the training corpus. This corpus treatment prompts the language model to learn the pattern of how characters take turns and to prompt itself on the learned pattern of character turn-taking during generation. However, in order to compare our system against that of Guan et al., we must enforce the telling of a two-character narrative in an interleaving fashion where characters take turns being the subject. While this is a simplifying adjustment, it allows for a fair comparison with the baseline. Appendix A.3 details specifically how this is accomplished.

3.2 Commonsense Inferences

To produce commonsense inferences for each sentence, we use the COMET model (Bosselut et al., 2019) to infer a set of ATOMIC (Sap et al., 2019) relations for each generated sentence. COMET generates commonsense inferences for a single sentence, referring to `PersonX` as the sentence's subject, and `Others` as other characters in the story. Due to our two-character closed-world assumption, we assume `Others` to refer to the second character.

We identify six of ATOMIC's nine relation types that are useful for creating coherent relations between story events: `xIntent`, `xNeed`, `xAttr`,

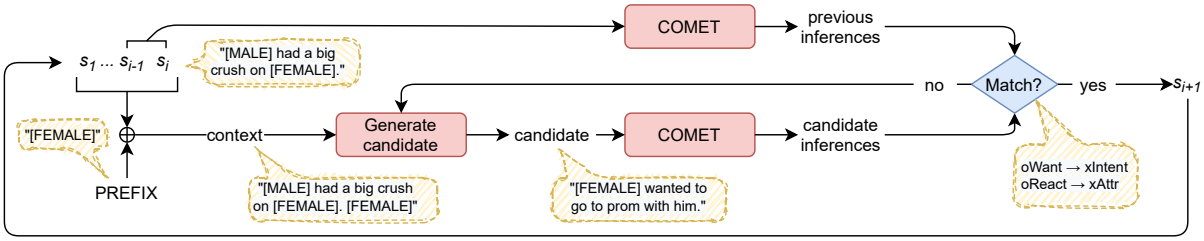

Figure 2: The overall procedure of generating two-character narratives with the CAST pipeline.

Type	Description
`xIntent`	The reason why `PersonX` would cause the event
`xNeed`	What `PersonX` might need to do before the event
`xAttr`	How `PersonX` might be described given the event
`oReact`	The reaction of `Others` to the event
`oEffect`	The effect the event has on `Others`
`oWant`	What `Others` may want to do after the event

Table 1: Definitions of the six types of ATOMIC relations that CAST uses. The top set describes preconditions of the current sentence and the bottom set describes postconditions of the previous sentence. Dotted lines show how we match postconditions to preconditions during the filtering phase.

`oReact`, `oEffect`, and `oWant`. Table 1 provides the definitions. We treat the first three relation types as *preconditions* of an event because they infer facts that might need to be true for an event to be enacted, such as a character having an intention, a goal, or a property. We categorize the final three as *postconditions* of an event because they infer things that might change once an event is enacted, such as a character reaction, an effect of the action, or a character forming a new intention.

Once we have inferred relations for the previous sentence s_{i-1} and current candidate c, we look for specific patterns between the sets of relations:

- The event in s_{i-1} affects the wants of a character, which manifests as an intention of the primary character in c (`oWant`→`xIntent`).
- An effect of the event in s_{i-1} manifests itself as something the primary character needs in c (`oEffect`→`xNeed`).
- A reaction to the event in s_{i-1} is expected and matches some property of the primary character in c (`oReact`→`xAttr`).

In this way, we create 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. The `oWant`→`xIntent` pattern corresponds to how readers track how characters form and carry out intentions in reader comprehension models (cf. (Graesser et al., 1991)). The `oEffect`→`xNeed` and `oReact`→`xAttr` correspond to how readers track causal consequences in reader comprehension models (cf. (Trabasso and Van Den Broek, 1985)).

To filter out "unqualified" sentence candidates generated by the language model, we match the inference types described in Table 1 and their arguments. For example, suppose sentence s_{i-1} is "`[MALE]` gives `[FEMALE]` a burger". Its `oEffect` is "have a burger". If sentence s_i is "`[FEMALE]` eats the burger", it will have an `xNeed` that is also "have a burger", which represents the prerequisite of "having a burger" in order to "eat a burger".

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. 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 lower-bound. Empirically, we find this value best considers the inferences listed above as matches, but excludes less-related inferences. Table 2 (top) shows how the threshold affects success rate—the percentage of queries that find a match within 100 generated candidates—and the diversity of results as measured by self-BLEU score (described in §4). Each system was conditioned on the same 30 2-character prompts from ROCStories with 5 different random seeds, requiring two of three inference type pairs to match to qualify as a match. Failure to

Semantic Similarity	# of Sentence Candidates	Success Rate	Self B-2	Self B-3
0.8	19.10	89.67%	.1426	.0552
0.85	26.54	82.33%	.1551	.0634
0.9	26.86	70.85%	.1709	.0796
0.95	34.00	63.50%	.1889	.1104
# of Matching	**# of Sentence Candidates**	**Success Rate**	**Self B-2**	**Self B-3**
1	2.32	100.00%	.1227	.0548
2	19.10	89.67%	.1426	.0552
3	104.63	29.54%	.1447	.0569

Table 2: Ablation study result for semantic similarity and required matching inference type pairs. *# 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 100-candidate limit. A higher Self-BLEU score (B-n) implies less diversity of the document (Zhu et al., 2018) (see §4).

Seed Prompt: Bob wanted to ask Alice out on a date.
Semantic Similarity = 0.8; # of matching = 2: Alice doesn't have a **date** to go on. Bob decided to ask her out. Alice is getting accepted for his **date**. Bob feels so happy and finally enjoyed his **date**.
Semantic Similarity = 0.95; # of matching = 2: Alice was nervous but agreed to **the date**. Bob prepared to go on **the date**. Alice enjoyed **the date**. Bob was happy **the date** went out of style.
Semantic Similarity = 0.8; # of matching = 1: Alice went out with me. Bob ordered some **cookies**. Alice liked the **cookies**. Bob is glad she went out with me.
Semantic Similarity = 0.8; # of matching = 3: Alice was scared to ask her out. Bob was nervous she would ask her out. Alice felt comfortable asking her out. Bob plans on getting on a date with her.

Table 3: Story examples generated by CAST with different semantic similarity thresholds and numbers of required matching inference type pairs. The story generated at 80% and 95% similarity (the first and the second story) both follow a single topic (**bolded**), but the first story maintains more diversity (underlined). Stories required to match one inference type pair mostly only match `oReact`→`xAttr` ("happy" emotion in the third example). Stories required to match three inference type pairs cannot find matches within candidate limit.

find a match within the candidates limit (100) will relax the matching constraints to one pair. Hence, the average number of sentences candidates might be over the candidate limit. As observed in Table 2 (top), 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 examples in Table 3).

In order to balance computation time and quality of the match, we only require two of three inference type pairs to match between a seed and a candidate sentence. When requiring three matches, CAST only finds a "qualified" sentence 29% of the time within 100 attempts (see Table 2 (bottom), computed at 0.8 semantic similarity). In practice (see examples in Table 3), we find requiring two pairs results in higher quality sentences than if we only require one out of three pairs to match, but is significantly more efficient than three out of three.

3.3 Fine-tuning on Commonense Inferences

CAST generates candidate sentences until one passes the inference matching process. Each candidate considered requires a set of commonsense inferences and semantic matching—computationally expensive processes. We ask whether it is possible to train a generative language model to emulate the heuristic CAST process, thereby producing coherent stories faster, with less computational overhead, and with less rigid constraints on its language production. In this section we fine-tune GPT-2 using a policy gradient reinforcement learning technique inspired by Peng et al. (2020), but using CAST to generate an exemplar dataset.

As CAST generates sentences, it stores sentence pairs along with a label: 0 for no relation matches and 1 for two or more matches. Some examples are given in Table 4. After generating a full story of k sentences, GPT-2 is fine-tuned on the labeled sentence pairs using the reinforcement learning technique. To punish the generation of "unqualified" sentences, we use the following loss function:

$$loss_{\mathrm{RL}}(s) = \begin{cases} loss_{\mathrm{s}}(s) & \text{if} \quad \text{label} = 1 \\ loss_{\mathrm{s}}(s) \times \rho & \text{if} \quad \text{label} = 0 \end{cases} \tag{1}$$

where $loss_{\mathrm{s}}(s)$ is the cross-entropy loss given by Radford et al. (2018) and ρ is a constant ($\rho \geq 1$) controlling the strength of punishment—sentences without matches incur more loss. The first sentence of the pair is masked so that loss is only calculated on the logits that produce the successor sentence.

Sentence Pair	Inferences
[MALE] had a big crush on [FEMALE]. [FEMALE] was constantly yelling at him.	None
[MALE] had a big crush on [FEMALE]. [FEMALE] wanted to go to prom with him.	oReact→xAttr oWant→xIntent
[MALE] took [FEMALE] fishing in the summer. [FEMALE] got very sick.	None
[MALE] took [FEMALE] fishing in the summer. [FEMALE] was loving it.	oReact→xAttr oWant→xIntent

Table 4: Examples of generated labeled sentence pairs. Sentence pairs with no matching inferences are labeled with 0, while those with two or more matching inference pairs are labeled with 1.

Seed Prompt: Bob invited Alice to hang out.
CAST: Alice planned a nice ***dinner*** for Bob. Bob and Alice spent all evening **cooking *dinner*** together. Alice was happy to see her ***dinner* cooked**. Bob was impressed with how **delicious** her ***dinner*** was.
GPT-ROC: Alice thought Bob was funny. Bob got mad and threatened Alice with punches. Alice ended up running away from Bob. Bob was awarded the fun they had together.
CAST-RL: Alice invited Bob to *hang out.* Bob agreed, and was happy to meet her. Alice was very happy and liked Bob. Bob and Alice still *hang out* after that.
Guan et al. (2020) Alice thought she would like it. Bob only wanted to show her the interest of a nerdy rlist. Alice seemed to like Bob asked Bob because she likes him. Bob ended up meeting Alice after school.

Table 5: Story examples generated by CAST, CAST-RL, GPT-ROC and Guan et al. (2020). The story generated by CAST follows a single topic (**bolded**) – cooking dinner, and shows a good plot coherence. The story generated by GPT-ROC fails to maintain plot coherence (underlined) during generation. CAST-RL generates relatively more repetitive/boring but logically coherent narrative (in *italic*). Guan et al. (2020) also suffers in plot coherence. More examples are given in Appendix B.

4 Experiments

We conducted two experiments to evaluate the CAST technique. The first experiment compares CAST to two unconstrained neural language model story generators: GPT-2 and the work by Guan et al. (2020) (§3.2). The second experiment assesses whether CAST can be used to fine-tune a neural language model (§3.3). Story examples can be found in Table 5 and Appendix B.

Metrics. We evaluate performance using human participant evaluation and automated metrics. Purdy et al. (2018) proposed a set of questions for evaluating story generation systems that includes dimensions such a logical coherence, loyalty to plot, and enjoyability. A variation of these questions have been used in evaluations of other story generation systems (cf. (Tambwekar et al., 2019; Ammanabrolu et al., 2020, 2021)). We modify a subset of these questions to simplify the language and focus more on participants' overall impressions of the narrative coherence:

- The story FOLLOWS A SINGLE TOPIC
- The story AVOIDS REPETITION
- The story uses INTERESTING LANGUAGE
- The story makes better LOGICAL SENSE

We conduct our studies on a crowdworking platform. Only those who pass a screening question are qualified for the study. Participants must also explain their preferences with more than 50 characters of writing. This helps filter out low-quality responses and ensures the validity of the study.

Because diversity of generated stories is important, we also measure 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. We define the self-BLEU score of the model to be the averaged BLEU score of its generated stories. A higher self-BLEU score implies less diversity of the stories. We report 2- and 3-gram self-BLEU scores (B-2 and B-3).

Baselines. We evaluate CAST against two baselines. The first is GPT-2-small fine-tuned on ROCStories (GPT-ROC), following prior work (Guan et al., 2020). The second baseline is the system by Guan et al. (2020), which fine-tunes GPT-2-small on ATOMIC and ROCStories using a multi-objective training procedure.

4.1 Experiment 1: CAST

We assess whether the CAST constraint method improves story coherence over the GPT-ROC and Guan et al. (2020) baselines. We recruited 116 participants on a crowdsourcing platform. Each participant read a randomly selected subset of 6 story pairs, comprised of one story from CAST and one from one of the baselines, GPT-ROC or Guan et al.. For each of the four questions listed above, participants answered which story best met the criteria.

Models	Story length	Logical Sense Win %	Lose %	Tie %	Single Topic Win %	Lose %	Tie %	Avoid Repetition Win %	Lose %	Tie %	Interesting Language Win %	Lose %	Tie %	Num resp.
CAST vs GPT-ROC	5	**53.7****	17.0	29.3	**52.4****	14.3	33.3	18.5	**25.3**	56.2	**30.6**	23.8	45.6	147
	10	**63.3****	19.3	17.4	**55.0****	14.7	30.3	33.9	**48.6**	17.5	34.9	34.9	30.2	109
CAST vs Guan et al.	5	**64.8****	19.4	15.8	**62.5****	18.5	19.0	26.9	**31.9**	41.2	27.4	**34.4**	38.2	216
	10	**73.9****	11.3	14.8	**86.7****	7.6	5.7	27.6	**50.5****	21.9	29.5	**48.6***	21.9	115
GPT-ROC-RL vs GPT-ROC	5	**68.1****	21.3	10.6	**63.8****	20.2	16.0	33.0	**39.4**	27.6	**41.5**	29.8	28.7	94
CAST-RL vs GPT-ROC-RL	5	42.7	42.7	14.6	**31.1**	25.6	43.3	26.7	**35.6**	37.7	24.4	**25.6**	50.0	89
CAST-RL vs CAST	5	**45.2**	38.1	16.7	**36.5**	31.0	32.5	27.2	**39.2**	33.6	23.2	**40.8***	36.0	126

Table 6: Human-participant evaluation results for experiments 2 and 3, 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 30 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.

Models	Length	B-2	B-3
GPT-ROC	5	.0749	.0251
	10	.1686	.0617
CAST	5	.1103	.0279
	10	.2017	.0883
Guan et al.	5	.0682	.0079
	10	.1773	.0520
GPT-ROC-RL	5	.2604	.1329
CAST-RL	5	.1912	.0949

Table 7: Self-BLEU n-gram (B-n) scores on 30 test-set prompts. Lower score indicates more diversity of generated stories.

Model	Train Set	Test Set
CAST	20.43 ± 3.40*	28.31 ± 3.39*
CAST-RL	8.72 ± 1.95*	11.06 ± 1.81*

Table 8: The average number of sentence candidates generated before finding a match. A confidence level of 95% over 5 random-seed runs for each model is presented. Each system was conditioned on the same 30 (training) and 10 (test) 2-character prompts from ROCStories. * indicates the difference between CAST and CAST-RL is statistically significant at $p < 0.01$ using the Mann-Whitney U test.

Participants read either 5-sentence or 10-sentence stories. The 10-sentence stories allow us to look at the effect of story length on coherence for CAST and the baseline GPT-ROC. To generate the stories, we again randomly selected 30 2-character prompts from the ROCStories corpus to seed the systems. We also calculated self-BLEU scores on these stories (Table 7).

The results are shown in Table 6 (top). The results indicate that our model performs significantly better than the GPT-ROC baseline on the "Logical Sense" and "Single Topic" dimensions. We conclude that CAST improves the coherence of generated narratives and stays more on topic, while retaining comparably interesting language and avoidance of repetition (neither of which are statistically significantly different from the baseline). When we increase the lengths of stories to 10 sentences from 5, a greater proportion of participants prefer CAST to the baseline in the dimensions of "Logical Sense" and "Single Topic" ($\sim 10\%$ increase). This shows that stories generated by CAST retain coherence longer relative to the baseline language model operating without constraints.

CAST also significantly outperforms Guan et al. in the "Logical Sense" and "Single Topic" dimensions. When increasing story length to 10 sentences, the coherence of stories generated by the Guan et al. system drops substantially compared to our model. Our model is worse at avoiding repetition and using interesting language, though these differences are only statisically significant on the 10-sentence stories. Table 7 also indicates the language diversity of baselines is higher than our model. We consider this a valuable trade-off for logical coherence.

4.2 Experiment 2: Reinforcement-Based Fine-Tuning

Having established that CAST improves coherence, we ask whether reinforcement-based fine-tuning (Section 3.3) can be used to teach a language model to model the CAST filtering method directly. In doing so we hope to (a) increase the robustness of the neural language model and (b) reduce the overall computation time by foregoing the inference and

semantic matching steps. We hypothesize that fine-tuning on CAST labels can increase robustness by generating continuations that will pass the filtering process with higher likelihood. This in turn can decrease the average number of generated sentence candidates required to find a match.

After generating 30 5-sentence stories with CAST using 40 different random seeds, we train GPT-ROC with the $2 \times 4 \times 30 \times 40$ generated labeled sentence pairs as described in Section 3.3.[1] This results in a new fine-tuned language model, which we refer to as GPT-ROC-RL. 40 different seeds for the language model are used to generate sentence pairs in order to simulate fine-tuning on very small datasets. Consistent with previous experiments, we use GPT-ROC as our baseline. In addition, we ask whether applying CAST filtering *on top of* GPT-ROC-RL further improves story coherence. CAST-RL is GPT-ROC-RL (that is, fine tuned on CAST labels) with CAST filtering applied to the outputs.

To evaluate whether fine-tuning on CAST labels improves story coherence, we recruited 59 participants and replicated the protocol in Section 4.1, except participants read pairs of stories from GPT-ROC, GPT-ROC-RL, CAST, or CAST-RL.

Table 6 (bottom half) shows the percentage of times stories from each system are preferred for each metric. GPT-ROC-RL improves coherence over GPT-ROC to approximately the same degree as CAST does over GPT-ROC, indicating that RL fine-tuning can be an alternative to constraint-based filtering on short (5-sentence) stories. CAST-RL does not significantly improve the logical sense of stories over GPT-ROC-RL, indicating that the reinforcement learning process has generalized the CAST inferential constraints into the language model. GPT-ROC-RL performs better on avoiding repetition and interesting language than CAST-RL in human evaluation, however not at a statistically significant level. Table 7 shows that GPT-ROC-RL performs worse on diversity than CAST-RL evaluated with the Self-BLEU metric.

Finally, CAST-RL improves the logical sense and single topic a small amount over CAST, but not at a statistically significant level. Despite being trained on CAST labels, the filtering process can still improve coherence. However, it does so at the expense of repetition and interesting language, which is also shown in Table 7.

Importantly, Table 8 shows that fine-tuning on CAST labels reduces the average number of candidates that need to be generated for a "match" to 11 from 28. The RL fine-tuning procedure reduces CAST's computation by 60.93%. Further fine-tuning results in an overfit language model but this model is sufficient to tie models explicitly using constraints (e.g., CAST-RL) on the issue of coherence. Tables 6 and 8 taken together show the benefit of CAST-RL for efficiency at no cost to desired model characteristics.

[1] We generate 2 "matched" sentence pairs during each sentence generation, one pair of each label. When $k = 5$, 4 sentences are generated to form a 5-sentence story.

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. We present a new approach to neural story generation that attempts to model the inferred event relations and character intentions of human readers in order to generate more coherent stories. The CAST method provides hard constraints to neural language model generation that results in greater story coherence, which crucially does not degenerate as the story gets longer. We further show that we can train neural language models to emulate the logic of the CAST constraints using reinforcement learning.

Our approach is relatively straightforward; the CAST method can be applied to any neural language model or be used to create a labeled fine-tuning dataset. The approach presents a first attempt at building text generation systems that reason about their reader along multiple dimensions.

References

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

Prithviraj Ammanabrolu, Ethan Tien, Wesley Cheung, Zhaochen Luo, William Ma, Lara J Martin, and Mark O Riedl. 2020. Story realization: Expanding plot events into sentences. In *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 ACL*, pages 4762–4779.

Faeze Brahman and Snigdha Chaturvedi. 2020. Modeling protagonist emotions for emotion-aware storytelling. In *Proceedings of EMNLP*, pages 5277–5294.

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.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th ACL*, pages 889–898.

Art Graesser, Kathy L. Lang, and Richard M. Roberts. 1991. Question answering in the context of stories. *Journal of Experimental Psychology: General*, 120(3):254–277.

Art Graesser, Murray Singer, and Tom Trabasso. 1994. Constructing inferences during narrative text comprehension. *Psychological Review*, 101(3):371–395.

Jian Guan, Fei Huang, Zhihao Zhao, Xiaoyan Zhu, and Minlie Huang. 2020. A knowledge-enhanced pretraining model for commonsense story generation. *Transactions of the ACL*, 8:93–108.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

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

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.

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.

Nanyun Peng, Marjan Ghazvininejad, Jonathan May, and Kevin Knight. 2018. Towards controllable story generation. In *Proceedings of the 1st Workshop on Storytelling*, pages 43–49.

Xiangyu Peng, Siyan Li, Spencer Frazier, and Mark Riedl. 2020. Reducing non-normative text generation from language models. In *Proceedings of the 13th INLG*, pages 374–383.

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, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey 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 EMNLP*, pages 4274–4295.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of EMNLP-IJCNLP*, pages 3982–3992.

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 AAAI*, volume 33, pages 3027–3035.

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*.

Tom Trabasso and Paul Van Den Broek. 1985. Causal thinking and the representation of narrative events. *Journal of memory and language*, 24(5):612–630.

Lili Yao, Nanyun Peng, Ralph Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-and-write: Towards better automatic storytelling. In *Proceedings of AAAI*, 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.

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.[2] We shuffle and split the ROCStories dataset into 80% (78,528) train and 20% (19,633) test sets.

Following Guan et al. (2020), character names in the ROCStories corpus are replaced with `[MALE]` or `[FEMALE]` 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 `[MALE]` and `[FEMALE]` 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 all 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. All models were trained on single GeForce RTX 2080 GPUs in Pytorch using the Huggingface Transformers library.[3] We replicate the multi-task baseline of Guan et al. in Tensorflow using their provided code.[4] 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 all models.

A.3 Character Conditioned Generation

We applied two methods to enforce the telling of a two-character narrative in an interleaving fashion wherein characters take turns being the subject of each sentence. The first method is to directly condition the language model on the concatenation of the previous story context and the tag denoting the character who is to take the next turn. For example if the last sentence in the story is "`[MALE]` was upset with `[FEMALE]`.", we append the tag of the opposite character to the context, i.e., `[FEMALE]`. Since the next sentence now starts with the tag associated with the opposite character of the one who acted in the previous sentence, the language model will be biased toward describing an action for that character.

The second method is to fine-tune the language model by formulating the input as $*\mathrm{T}*[s_1, \ldots, s_{i-1}]$, where T is the tag denoting the character who is to take a turn. For example, for the story, "`[MALE]` was upset with `[FEMALE]`. Because of this, `[FEMALE]` apologized.", the prompt is formulated as "* `[FEMALE]` * `[MALE]` was upset with `[FEMALE]`." The language model is fine-tuned by back-propagating the loss calculated on the sentence "Because of this, `[FEMALE]` apologized."

A.4 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[5]) 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 2 of 3 inferences to match. Runtime is feasible due to matching on two out of three 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.

A.5 Preliminary Study Questions

- The story FOLLOWS A SINGLE PLOT.
- The story's events OCCUR IN A PLAUSIBLE ORDER.
- The story AVOIDS REPETITION.

[2] https://cloud.tsinghua.edu.cn/d/670f7787b6554f308226/
[3] https://huggingface.co/transformers/
[4] https://github.com/thu-coai/CommonsenseStoryGen
[5] https://mosaickg.apps.allenai.org/comet_atomic

- The story IS ENJOYABLE.

A.6 RL Fine-Tuning

We generate 30 5-sentence stories with CAST, using 40 different seeds. 2 sentence pairs are generated when generating each following sentence and each 5-sentence story is composed of 4 following sentence. Hence, $2 \times 4 \times 30 \times 40$ sentence pairs are generated to fine-tune GPT-ROC with reinforcement learning technique for 1 iteration. We use $\rho = 0.001$ in Section 3.3 to punish the "unmatch" sentence pairs.

B 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.

GPT-ROC:
Alice was being very loud.
Bob told Alice they did not agree.
Alice ordered pizza to be delivered.
Bob showed up later and Alice was pleased.

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.

GPT-ROC:
Alice said yes.
Bob put on his clothing.
Alice walked out of the restaurant.
Bob didn't give Alice her favorite shirt.

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.

GPT-ROC:
Alice took his friend's boat.
Bob brought with him to the water.
Alice stuck her hand out a river but the fishing got stuck.
Bob snapped his fishing line and the boat stopped in the middle.

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.

GPT-ROC:
Alice invited him to her wedding.
Bob refused and immediately broke down.
Alice stormed out of the door.
Bob was very upset that he would never see her again.

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.

GPT-ROC:
Alice really needed Bob to come with her.
Bob knew he couldn't get her to come with him.
Alice saw Bob grow up and change his attitude.
Bob is grounded because Bob is getting away with everything.

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.

GPT-ROC:
Alice asked him to meet her at the restaurant.
Bob refused to get her out of the car.
Alice began to panic and began to argue with Bob.
Bob ended up never talking to Alice.
Alice began to return the phone and she called the restaurant.
Bob got a new date with Alice.
Alice left Bob with a very happy reunion.
Bob is so happy he is now dating Alice.
Alice is so happy she loves her 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 though, 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.

GPT-ROC:
Alice had a fake boyfriend.
Bob found out Alice was missing.
Alice promised to find her.
Bob found her in an alley.
Alice never broke up with Bob again.
Bob never broke up with Alice.
Alice was so happy to see him again.
Bob is very happy being close to her.
Alice is happy that he's okay now.

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.

GPT-ROC:
Alice wanted to buy her a necklace.
Bob felt bad for her.
Alice got the necklace.
Bob made it.
Alice was very happy with the necklace.
Bob was glad that she got to see Alice.
Alice will never forget her great necklace.
Bob will never forget the necklace that Alice bought.
Alice is glad that she bought her necklace.

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 tattooy 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.

GPT-ROC:
Alice and Bob played soccer.
Bob's mom told him to throw a kick.
Alice kicked him in the leg.
Bob was glad that he never played there again. Alice's mom apologized for the incident.
Bob's mom never learned to kick in the leg again.
Alice was very glad he never stopped playing soccer with her.
Bob was happy that he was never kicked again.
Alice still still plays soccer every now and loves hertime.

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.
