Fine-to-Coarse Entailment Hierarchy Construction for Coarse-to-Fine Story Generation

Haw-Shiuan Chang **Nanyun Peng** **Mohit Bansal** **Tagyoung Chung**
Amazon AGI Foundations
{chawshiu,pengnany,mobansal,tagyoung}@amazon.com

Abstract

When users want to write a story with a language model (LM) assistant such as ChatGPT, it is often very difficult to provide a prompt that clearly specifies all their interests. For the providers of LM assistants, it is also difficult to ensure their output stories come from a dataset without copyright concerns. Motivated by these limitations, we propose a coarse-to-fine (C2F) tree-based story generation framework, which is called C2F-StoryTree, where the LM iteratively generates more and more specific story prompts based on a user's input prompt and the desired plot selected by the user. To realize our C2F-StoryTree framework, we propose an entailment hierarchy (EH) text structure, in which a more specific response entails more general prompt (e.g., a story entails a summary). We also propose novel annotation tasks, decoding methods, and a human-and-machine-in-the-loop procedure to minimize the annotation cost of building the text structure. We build an entailment hierarchy dataset on top of the story datasets with desired licenses and styles, on which the service providers can fine-tune or evaluate their LMs. In our experiments, we demonstrate that our C2F-StoryTree system not only allows users to collaborate with a LM in a new paradigm, but also generates the short stories with human-level quality. Furthermore, our entailment hierarchy and C2F generation substantially make the generated stories more diverse, coherent, engaging, and creative.

1 Introduction

As (large) language models (LMs or LLMs) become more and more powerful, many users have started using an LM to assist their creative writing tasks, so improving the user experience of writing stories with LM is more important than ever. Typically, a user would provide the LM with a prompt that specifies their intent, read the generated stories, and modify the prompt accordingly in a dialog if they are unsatisfied with the generation outputs (Xu et al., 2018; Peng et al., 2018; See et al., 2019; Zhong et al., 2023; Xie et al., 2023; Yao et al., 2023a). However, this typical workflow is often not satisfactory to users for several reasons. First, it usually takes lots of effort to come up with a good prompt (Yuan et al., 2022; Mishra and Nouri, 2022) because users often only have a vague idea about the stories they want to write. Second, since the models generate the full story directly, users lack an intuitive way to control the main plot of the story. Finally, generating many long stories could be computationally expensive, and to a user, reading these undesired long stories again and again is highly cognitively demanding.

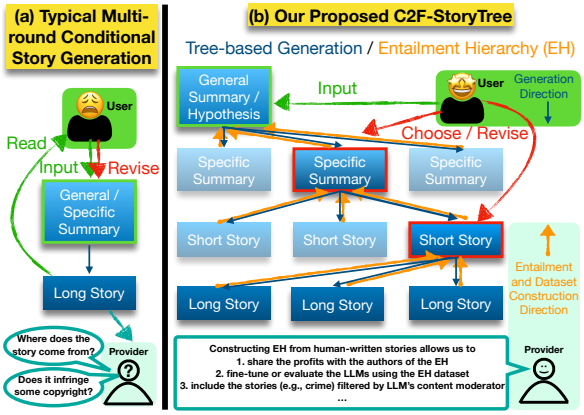


Figure 1: (a) In the typical framework, users often have to repeat the cycle of reading the undesired stories and modifying the prompts in a dialog and the service providers often cannot reduce the risks of infringing copyrights. (b) In our C2F-StoryTree framework, a user can write a general summary/hypothesis, and then our LM provides a more specific summary. The user can choose the desired specific summary as the prompt for the short story, and so on to iteratively expand the story. To realize this framework and give providers more control, we build an entailment hierarchy (EH) dataset and train an LM at each layer to iteratively expand the story prompt.

Besides users, the service providers also lack control over LLM generations. For providers, it is hard to identify the exact source of each output

General Summary/Hypothesis as the Input Prompt: "A girl achieved something impressive."

Short Story from CG (Conditional Generation): "A girl was very tall with a long teeter totter. But she wanted to do something with it. Her parents bought her a teeter totter. The girl sat on the stool and stared at it intently. She had finally completed something that would be impressive to her mom."

Stories from EH + rerank (T5-3b FT)

Specific Summary from T5-3b^{h2s}		**Short Story** from T5-3b^{s2s}		**Long Story** from T5-3b^{s2l}
Text	RS	Text	RS	Text
1. A young girl did something extraordinary during the school year.	0.79	1. Amy had a test on Friday. She was very scared. She decided to jump rope. Amy threw the rope very hard. Amy got a straight A.	0.99	1. Amy was a nerd, a nerdy teen. ... So when she heard about the test she was scared. She figured, she would have to jump rope. ... She threw the rope and she got a straight A! She was so happy!
2. A girl had a great achievement on her birthday.	0.77			
3. A girl has become a runner after running a half marathon.	0.73			
4. A girl did something impressive at the end of the day.	0.65	2. A gymnast went to her gymnastics team practice. ... She got on the trampoline and started to flip. ... Her coach was proud.	0.98	
5. A student made an impressive effort to reach her goal.	0.25			
6. Someone had an experience with achieving something at school.	0.03			

Table 1: An example generated by our C2F-StoryTree framework and the conditional generation baseline. We highlight the selected text for generating more specific text. RS refers to the reranker scores.

story. Therefore, we do not know if the story is protected by copyright (Ippolito et al., 2022a; Chang et al., 2023; Min et al., 2023; Reisner, 2023) and we cannot share our profits with the authors of the original story. Moreover, it is also difficult to remove the undesired stories, increase the diversity of stories (Jentzsch and Kersting, 2023; Eldan and Li, 2023), or include some stories that are filtered by LLM's content moderator (e.g., crime stories).

Motivated by users' dissatisfactory experience and providers' controllability requirements, we propose C2F-StoryTree, a coarse-to-fine tree-based generation framework that iteratively increases the specificity and length of the story prompt. In the example of Table 1, a user inputs "*A girl achieved something impressive*" as the prompt. The typical conditional generation baseline would directly output a short story. In contrast, C2F-StoryTree first provides several more specific prompts. Assuming the user chose the highlighted option, our system generates multiple short stories. The user can further select the highlighted short story to create longer stories. Using C2F-StoryTree, the human and machine can work together to iteratively build a search tree structure from coarse to fine.

In Figure 1, we can see the coarse-to-fine generation process leads to a hierarchical structure. We assume each prompt is implied (entailed) by multiple more specific prompts[1] (i.e., each node in the upper layer is entailed by multiple nodes in the lower layer), so we call the text data structure an 'entailment hierarchy' (EH). We use crowdsourcing to build an EH dataset, which consists of thousands of entailment labels between the text layers, and show that this new dataset helps the service providers enhance or evaluate an LLM while minimizing the copyright concerns. In short, the proposed C2F-StoryTree framework gives both users and providers more control on the story generation.

In this study, we focus on minimizing the cost of constructing the EH dataset to maximize our impact. We propose a novel fine-to-coarse EH construction method to realize the coarse-to-fine story generation framework: First, we identify a dataset containing many desired stories, which allows us to negotiate the profit sharing with the authors of the stories. Second, we construct the EH dataset from bottom to top. As shown in Figure 1 (b), we collect the common summaries of two similar stories in the dataset (i.e., generating one parent node that is entailed by two children nodes) and use similar methods to construct the other upper layers. To further reduce the cost of generating the common summaries, we propose an EH construction method that leverages a novel decoder, reranker, crowdsourcing, and human-and-machine-in-the-loop techniques. Finally, we realize the C2F-StoryTree framework by training seq2seq models on the EH dataset that generate lower-layer nodes given an upper-layer node (e.g., generating short stories based on a specific summary).

In our experiments, we evaluate our EH construction methods and compare the story generation systems with or without the EH. The results show that the proposed method built on T5-3b (Raffel et al., 2020) increases GPT3.5's probability of successfully generating the common summary (from 45.5% to 72%). We also show that our EH built on ROC stories (Mostafazadeh et al., 2016) could significantly improve the short stories generated by T5-3b or Vicuna 7B (Chiang et al., 2023) in most automatic and human evaluation metrics (e.g., engagement and creativity). Furthermore, our common summary writing task naturally provides a specificity measurement of our prompt, which lets us discover that LLMs perform very differently on story generation given the prompts with different

[1]To reduce the scope of this work, we assume the prompts do not contain format constraints (e.g., the story must have some keywords or length).

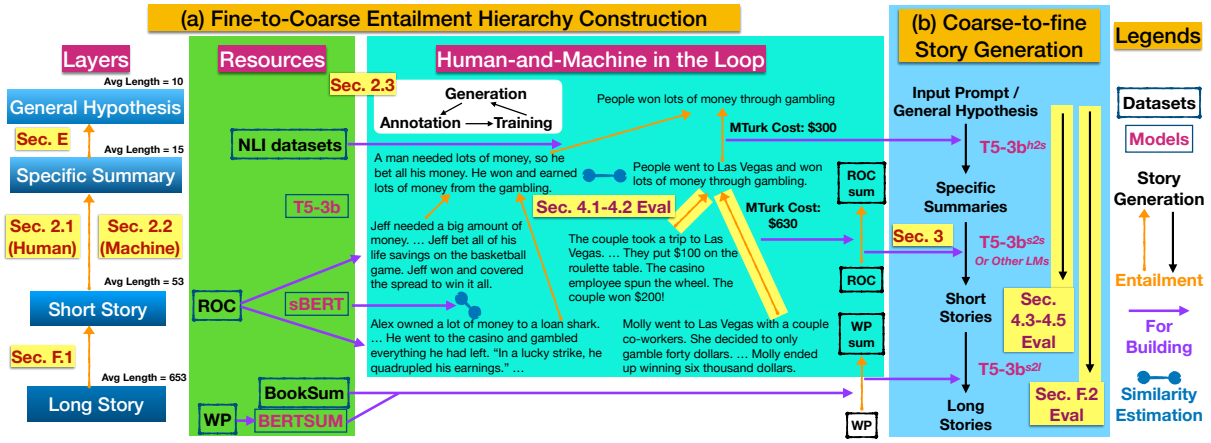


Figure 2: The realization of C2F-StoryTree in this work. (a) We construct the entailment hierarchy (EH) by leveraging existing resources and machine-and-human-in-the-loop techniques. After fine-tuning the models using the crowdsourced EH, we generate summaries for ROC and WritingPrompts (WP). (b) We train seq2seq models to generate lower-layer text from upper-layer text and achieve the coarse-to-fine (C2F) generation.

specificity levels.

2 Entailment Hierarchy Construction

As illustrated in Figure 2 (a), we summarize each lower node into the upper node in the fine-to-coarse direction. The fine-to-coarse construction is much more cost-efficient than the coarse-to-fine manual story writing because it is very expensive to hire humans who can write diverse good stories given a prompt (Li et al., 2023a). In contrast, writing the summaries for a story in an existing dataset could be easily done by a crowdsource worker.

In the following subsections, we describe our method of building the specific summary layer given a short story dataset. From left to right of Figure 2, we first describe how we ask humans (crowdworkers) to write the specific summaries given the stories in Section 2.1. Next, we describe how a machine (LM) generates the summary using a novel decoding method and reranker in Section 2.2. In Section 2.3, we leverage the annotations from humans, the predictions from machines, and the existing resources to minimize the cost of building the specific summary layer.

Our methods of constructing the other two upper layers are similar to the method described in this section. Due to the space restriction, we describe the adjustments of our construction methods in Appendix E and Appendix F.

2.1 Common Summary Writing Task

We want to build an entailment hierarchy (EH) on top of a pool of stories with the desired style and licenses. To ensure the quality of the EH, we first hire humans to build an upper layer based on the lower layer (e.g., writing their summaries that are entailed by the stories).

In Figure 3 (a), we compare the different options for establishing entailment relation between layers. The method on the right side is a standard option: asking annotators to summarize each story in a dataset. However, using this method to build the entailment hierarchy is problematic. The summaries written by workers could be too specific or too general, which makes the resulting hierarchy too deep or too shallow, respectively. A too shallow tree cannot allow users to iteratively specify their interests while a very deep tree is more expensive to construct and may require user's selections too many times. The standard story writing given the prompt on the left side of Figure 3 (a) also has this problem. The human writers need to carefully control the specificity of their summaries/stories so that the next prompt would become significantly more specific than the last prompt, while not becoming too specific to generate the diverse stories in the lower layer.

To reduce the cost and better control the specificity, we propose a common summary writing task. We ask crowd workers to write a common summary given two similar short stories. A common summary is a statement that is a summary of every single child node in the entailment hierarchy while being as specific as possible. The common summary writing task substantially reduces the specificity ambiguity, makes the answer more verifiable,

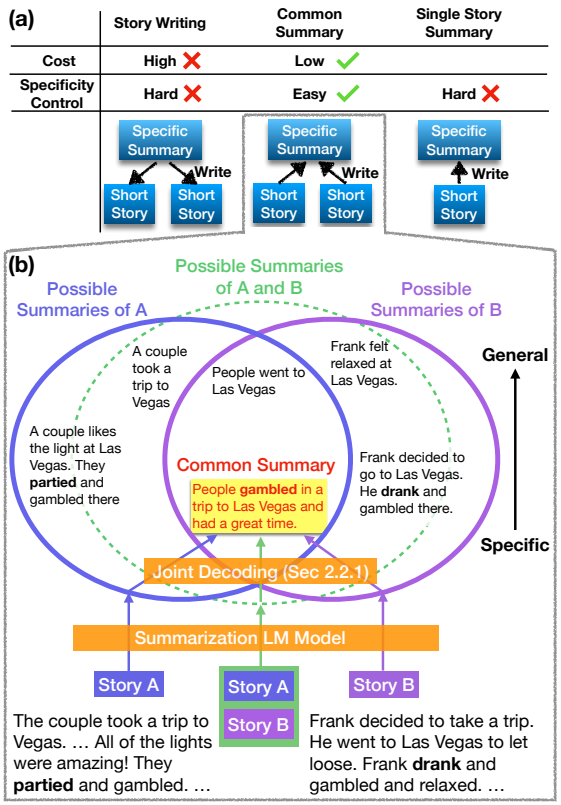


Figure 3: (a) Comparison of options for constructing the entailment hierarchy. (b) The task and our model of generating a common summary. We input story A, story B, and their concatenation into a summarization model and during the decoding time, we merge their generation probabilities word by word in order to get the common summary of both story A and story B.

and increases the number of free story-summary pairs we can derive from the transitive closure.

In Figure 3 (b), we visualize the two sets of possible summaries of two stories and their common summary in the intersection of the two sets. When a worker is asked to just summarize story A, the worker is allowed to provide a very specific summary, a very general summary, or a very trivial summary (e.g., copy the first sentence) in the big blue circle. In contrast, in our common summary task, the workers cannot write too specific summaries that cannot be entailed by both stories. The workers are also incentivized to provide more specific summaries to prevent their response from being rejected due to the instruction violation. Furthermore, the workers often need to summarize the stories creatively to increase the specificity (e.g., the common summary in Figure 3 (b) mentioned *people had a great time*, which is not explicitly mentioned in both stories).

2.2 Common Summary Generation

To generate the common summary of stories A and B, we train a seq2seq LM model that can summarize a single story and a pair of stories.

2.2.1 Joint Decoding with Concatenation

During testing, our goal is to generate a statement that is a likely output of the LM given story A (i.e., likely summary of story A), a likely output given story B, and a likely output given their concatenation simultaneously. In the example of Figure 3 (b), given the common context "*People ...*", the LM should not output *partied* (unlikely for story B) and *drank* (unlikely for story A). Instead, the LM should output *gambled* as the next word because "*People gambled ...*" could be a summary of story A and a summary of story B. Since each word in the common summary should not have a low generation probability given each of the three different inputs, we average the logits of their probabilities during the decoding. That is, the probability of the next word x at time $t+1$ is computed by

$$P_{t+1}(x|S) = \frac{\exp(\sum_{s\in S}(\boldsymbol{h}_s)^T \boldsymbol{w}_x)}{\sum_{x'}\exp(\sum_{s\in S}(\boldsymbol{h}_s)^T \boldsymbol{w}_{x'})}, \quad (1)$$

where the input set $S = \{A \oplus c_t,\ B \oplus c_t,\ A \oplus B \oplus c_t\}$ are the concatenation of each story and the current decoded context c_t, $\boldsymbol{h}_s$ is the hidden state of the input s, and $\boldsymbol{w}_x$ is the word embedding of x in the output softmax layer.

Using this decoding method, we can leverage the ability of an existing pretrained model to summarize a single story and the partially correct crowdsourced statements that summarize only one of the stories. This extra training data is especially helpful when we only have a few crowdsourced summaries in the beginning.

2.2.2 Reranker

We use top-k sampling (Fan et al., 2018) to generate several candidate summaries and use a reranker to filter out the ones that are less likely to be implied by the input stories, which has been shown to be very effective in summarization tasks (Ravaut et al., 2022). The reranker is a cross-encoder that takes a story and a summary candidate as its input and predicts the likelihood that the story entails the candidate. The final score of a candidate common summary is the minimum of its implication likelihoods from story A and from story B. The reranker is pretrained using the existing summarization dataset(s) and fine-tuned using our crowdsourced labels.

2.3 Data Collection using Human-and-Machine in the Loop

First, we ask workers to write common summaries and other workers to verify their answers. After collecting a few hundred summaries, we train an LM and a reranker to provide the summary candidates so that the worker just needs to verify the generated candidates. Verifying the generated summaries saves lots of writing effort and cost (Clark et al., 2018; Bartolo et al., 2022; Liu et al., 2022), provides many valuable negative examples to train the reranker, and could be used to evaluate the generated common summaries.

We use a T5-3b (Raffel et al., 2020), which has been pretrained for summarization tasks, as our seq2seq LM to generate a common summary of the two input stories. We choose DeBERTaV3-large (He et al., 2021a,b) as our reranker model and pretrain it using XSum (Narayan et al., 2018), a large summarization dataset that includes news from various domains.

We use Amazon Mechanical Turk (MTurk) to conduct crowdsourcing. Every written summary is verified by other workers to ensure good data quality and exclude inadequate workers. We randomly sample ROC stories (Mostafazadeh et al., 2016) with the top 2 highest sBERT similarities (measured by all-mpnet-base-v2 (Reimers and Gurevych, 2019)). Next, we ask workers to write and verify their common summaries.

Initially, we need to spend around $2.6 in the MTurk tasks to receive 4 entailment labels and 3.3 labels are positive. We find that around 300 common summaries are enough to train a high-quality generator. Then, the cost of getting 10.9 positive labels among 20 labels is reduced to around $1.5 in MTurk verification-only tasks. In total, we spend $630 US dollars to collect 3442 entailment labels between 1684 summaries and 497 stories, among which 2268 pairs are positive examples.

3 Coarse-to-fine Story Generation

The quality of the story generator depends heavily on the training set size, but many stories in our dataset do not have summaries. To augment the training data, we generate three summaries of each story in the ROC dataset. Then, we train T5-3b^{s2s} that outputs the short stories given each of their generated summaries as shown in Figure 2 (b). Similarly, we train T5-3b^{h2s} to generate the specific summaries after collecting their corresponding general hypothesis (see Appendix E for details) and train T5-3b^{s2l} to generate the long stories based on the short stories (see Appendix F).

4 Experiments

In the following subsections, we evaluate our generated common summaries given two short stories and evaluate the generated short stories given a common summary as shown in Figure 2. Please see more experiments in Appendix D and more details of our evaluation process in Appendix H.

4.1 Common Summary Generation Setup

To evaluate the decoding method and reranker proposed in Section 2.2, we randomly sample 100 story pairs in the ROC dataset that are unseen in our training data, and compare the generated common summaries from each method.

Common Summary Generation Methods: We compare our generation LM and reranker with GPT3.5 in-context learning. Then, we conduct an ablation study on our decoding methods in Figure 3 and Section 2.2.1. Joint decoding (**JD**) removes the reranker and the concatenation of story A and B. **DynE** (Hokamp et al., 2020) is the same as **JD** except that we average their probabilities rather than logits in Equation 1. Joint decoding with concatenation and reranker (**JDC + rerank**) is our proposed method. All our models are fine-tuned using the entailment relation from short stories to the specific summaries we collected. We augment our training dataset by also including the entailment relation from short stories to the general summary/hypothesis through transitive closure. (e.g., A $\rightarrow$ S $\rightarrow$ G implies A $\rightarrow$ G) and call the resulting model **JDC + rerank + hypo**.

Metrics: We automatically measure the fluency using the perplexity of GPT-2 XL (Radford et al., 2019) and the diversity using the ratio of unique n-gram (dist-n) (Li et al., 2016). In our human evaluation, the workers label a 1-5 score for fluency, how likely the generated text actually summarizes both stories (Entail 2), the probability of summarizing each story (Entail 1), and how likely it could summarize another story similar to story A (Specificity). If the generated summary is too general (e.g., *The story is about an accident*), the summary might be entailed by all the stories sharing similar topics. Contrarily, if the generated summary is specific enough, the probability of summarizing the story similar to an input story would be low.

Method	Model Size	Training Data Size: Specific Summary	Training Data Size: General Hypothesis	Automatic Metrics: len	Fluency: ppl (↓)	Diversity (%): dist-1	Diversity (%): dist-2	Human Judgement: Entail-2 (%)	Entail-1 (%)	Fluency	Specificity (↓) (%)
GPT3.5 (text-davinci-003) 1 Shot	175B	1		19.27	9.51	36.86	64.90	44	66.3	4.58	**23.5**
GPT3.5 (text-davinci-003) 5 Shot		5		15.87	**9.05**	37.98	**69.17**	45.5	66.5	**4.69**	25.5
DynE (Hokamp et al., 2020)	3B	3.4k		11.56	10.08	**41.41**	67.04	25	35	4.45	26.5
JD		3.4k		12.16	10.01	41.20	66.85	25	37	4.53	24
JDC		3.4k		11.05	10.19	41.04	65.32	33	47	4.53	26.5
JDC + rerank		3.4k		10.64	10.24	33.18	54.17	58	69.5	4.66	38
JDC + rerank + hypo		3.4k	1.9k	10.04	10.30	34.29	54.89	**72**	**80.5**	4.68	48

Table 2: Comparison of common summary generation methods. We use T5-3b models trained by the entailment hierarchy in dynamic ensemble (DynE) (Hokamp et al., 2020), joint decoding without concatenation (JD), joint decoding with concatenation using Equation 1 (JDC), our method with reranker (JDC + rerank), and our method trained by all summary/hypothesis layers (JDC + rerank + hypo). Entail n means the probability that the summary is implied by n stories. len is story length. ↓ means lower is better and the best scores are highlighted.

4.2 Common Summary Generation Results

The results of Table 2 show that every method can generate fluent summaries. **JDC + rerank** and **JDC + rerank + hypo** have a much higher Entail 2 (i.e., its generated common summary is more likely to be entailed by both input stories) compared to GPT3.5 while GPT3.5's common summaries are more specific. The lower Entail 2 of other baselines demonstrate the effectiveness of our proposed components, especially the reranker.

4.3 Short Story Generation Setup

One author of this paper wrote 60 common summaries/hypotheses for the pairs of stories in ROC dataset (Mostafazadeh et al., 2016) with various similarities. We treat the summaries/hypotheses as the user's prompt and one of the two stories as the reference story. We assume the user wants to have the output stories similar to the reference story. In Appendix F.2, we also compare the quality of the generated long stories.

Short Story Generation Methods: The conditional generation (**CG**) baseline directly outputs the story given the prompt, and the other methods use coarse-to-fine (C2F) generation. At each generation iteration, **EH** (entailment hierarchy) selects the prompts randomly, and **EH + rerank** selects the prompts using our reranker. To simulate the user's selection preference, **EH + rerank + sim** selects the prompts according to both the reranker scores and sBERT similarities to the reference stories.

To show the generality of our framework, we report the performance of replacing T5-3b^{s2s} with GPT-J 6B (Wang and Komatsuzaki, 2021) and Vicuna 7B v1.3 (Chiang et al., 2023). Similarly, we replace the T5-3b model in the **CG** baseline with LLM using 5 shot in-context learning including Vicuna 7B, GPT3 175B, and GPT3.5 175B. To know our upper bounds, we report the **ROC NN stories** baseline, which retrieves the three human-written stories in the ROC dataset that have the highest sBERT similarities with the reference story.

Metrics: In automatic evaluation, we compare the similarity to the reference story and the similarity to the input hypothesis. The relevant but creative output stories should have high reference similarity but low input similarity. We provide three automatic similarity measurements: ROUGE 1 F1 scores (Lin, 2004) (R1), R2, and sBERT similarity scores. To compare the generation diversity, we measure the dist-1 and dist-2 of three generated stories from each method given each summary/hypothesis.

In human evaluation, we report the results of two evaluation rounds. One for open-source models and one for the other models. Each round is done by different sets of MTurk workers, so they are not directly comparable. The workers judge if the generated story entails the input summaries/hypotheses, and give a 1-5 score to various quality metrics, including relevancy, creativity, coherence, and engagement.

4.4 Short Story Generation Results

Round Results of the Open-Source Models: Table 3 shows that **EH + rerank** significantly outperforms the conditional generation (**CG**) baseline in every human judgement dimension. **EH + rerank + sim** boosts the sBERT similarity score of **EH + rerank**, which demonstrates that our C2F methods could indeed faithfully follow the selection preference a user posted on the intermediate iteration.

In Table 1, we qualitatively compare their generated stories. **CG** often follows the prompt literally

Method	Story Generator	C2F	Automatic Metric: len	Reference Relevancy (%): R1	R2	sim	Creativity (%): R1 (↓)	R2 (↓)	sim (↓)	Coherence: ppl (↓)	Diversity (%): dist-1	dist-2	Human Judgement: Rel	Pro (%)	Cr	Coh	Eng
Round for Generators with Open-Source Licenses																	
CG	T5-3b FT		46.61	25.20	4.11	45.49	24.48	11.92	53.56	10.00	37.73	73.03	2.44	83.33	2.82	3.22	2.72
EH		V	47.94	23.70	3.03	43.51	**14.78**	**2.90**	**44.12**	**9.97**	**42.61**	**81.63**	2.48	71.67	**3.11**	3.81	2.88
EH + rerank		V	47.70	25.07	3.77	46.64	16.79	3.98	48.61	**9.97**	40.63	78.66	2.68	**92.50**	3.10	**3.88**	**2.97**
EH + rerank + sim		V	47.23	**25.71**	**4.25**	**49.27**	17.62	4.41	48.96	9.99	39.28	76.92	**2.76**	91.67	3.08	3.87	2.78
EH	GPT-J 6B FT	V	64.00	22.90	2.77	40.24	12.50	2.28	38.07	9.34	41.73	81.89	2.37	46.67	3.51	3.83	3.50
Round for Generators using Proprietary Data																	
CG	Vicuna 7B 5 Shot		46.21	23.77	3.21	43.79	20.68	6.98	50.74	10.01	**42.54**	80.47	2.55	**88.33**	3.10	3.99	3.06
EH + rerank	Vicuna 7B 5 Shot	V	49.85	**24.67**	3.56	45.65	17.74	4.16	50.59	9.86	41.43	80.83	2.59	86.67	3.28	**4.16**	**3.38**
EH + rerank	Vicuna 7B FT	V	52.33	24.43	**3.68**	**45.72**	**15.56**	**3.32**	**46.39**	**9.79**	42.37	**81.54**	**2.72**	80.00	**3.56**	3.98	3.36
CG	GPT3 175B 5 Shot		58.37	25.15	3.46	47.57	18.43	5.50	54.82	9.55	38.93	76.97	2.95	90.00	3.42	4.13	3.59
	GPT3.5 175B 5 Shot		57.91	24.94	3.46	47.26	18.49	5.37	54.41	9.57	39.71	78.26	3.02	90.00	3.67	4.35	3.63
ROC NN Stories	Human		50.87	27.75	4.50	61.56	11.87	1.42	36.95	9.86	46.38	86.47	2.78	54.17	3.52	4.20	3.58

Table 3: Comparison of generated short stories. Our coarse-to-fine (C2F) generation framework selects the generated option that has a high reranker score (EH + rerank) and our main baseline is a conditional generator without using entailment hierarchy (CG). FT means fine-tuning, Rn is ROUGE n F1, and sim is the similarity measured by sBERT. We report human judgments on the reference relevancy (Rel), prompt following/entailment probability (Pro), creativity (Cr), coherency (Coh), and engagement (Eng), which are not directly comparable across the two rounds. The best scores using a T5-3b or Vicuna 7B model are highlighted. GPT3 refers to *davinci* and GPT3.5 refers to *text-davinci-003*.

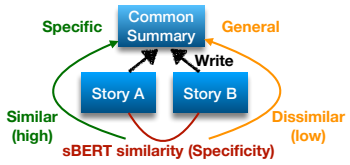


Figure 4: sBERT similarity between story A and B is a specificity measurement of their common summary.

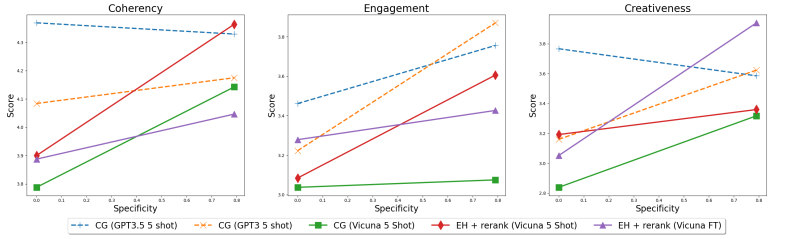


Figure 5: The linear trendlines of the short story scores from different LLMs given the prompt with different specificity levels.

by copying some words from the input prompt. This degrades not only the generation's creativity and diversity but also the coherence because the training dataset might not have any story explicitly mentioning the prompt. In contrast, **EH + rerank** could generate the stories that implicitly entail the input prompt mainly because our common summary task encourages the workers and LMs to generate highly abstract summaries.

Round Results of the Proprietary-Data Models: Training on copyrighted stories induces many legal risks (Reisner, 2023; Min et al., 2023), especially for commercial usage. Nevertheless, we still evaluate our framework on the LLMs that are pretrained on many professionally-written stories (Reisner, 2023). We found that our entailment hierarchy on top of crowd-written stories (**EH + rerank**) could still significantly improve LLM (Vicuna 7B 5 Shot) in terms of creativity, coherency, and engagement.

Compared to GPT3.5, the similar GPT3 scores suggest those generated high-quality stories may come from its pretraining corpus and its large size (Carlini et al., 2022). Compared to **CG (GPT3.5 175B)**, our **EH + rerank (Vicuna 7B 5 Shot)** achieves much better diversity, length control, controllability from the C2F process, and automatic creativity scores (i.e., fewer copies from prompts) using a much smaller LM size, but still do worse in relevancy, coherence, and engagement. Finally, the quality of **EH + rerank (Vicuna FT)** has already been close to the quality of **ROC NN stories** (i.e., human-written stories in our training data). Only its coherence and engagement are around 0.2 behind our training stories.

4.5 Analyses using Prompt Specificity Levels

As illustrated in Figure 4, we use the similarity to measure the specificity of the summary/hypothesis, and the measurement allows us to compare the story quality changes versus the specificity level of the prompt. This new evaluation dimension gives us several novel insights into the short story generation ability of LLMs and our methods.

First, Figure 5 shows that the stories from GPT3 get better for more specific prompts while GPT3.5 does better given more general prompts, probably because SFT and RLHF (Ouyang et al., 2022) filter

out the low-quality stories for the general prompt while reducing the number of valid stories for the specific prompt (Florian et al., 2024). Second, compared to 5-shot learning, fine-tuning Vicuna 7B using our dataset significantly boosts creativity due to our abstract common summaries and plenty of training stories. For the specific prompt, its creativity is even better than GPT3.5 and GPT3 significantly. High creativity and diversity are often the most desirable properties for some writers seeking new ideas (Ippolito et al., 2022b). As far as we know, this is the first study that demonstrates the feasibility of significantly surpassing GPT3.5 for this purpose using a much smaller LM size and the open-source crowd-written stories.

5 Related Work

Similar to our work, several studies (Drissi and Kalita, 2018; Chen et al., 2019; Yang et al., 2022a; Li et al., 2023b; Zhong et al., 2023; Yang et al., 2022b; You et al., 2023; Lee et al., 2024) advocate using summaries/outlines to control the story generation. In their work, the mapping between story and summaries could come from existing summarization datasets, off-the-shelf summarizers, or the power of LLM, while we focus on cost-effectively building the summarization datasets/models at multiple granularities to support the coarse-to-fine story generation.

Multi-granularity summary or reasoning process of LM could also be constructed automatically. For example, Xu et al. (2018) extract an event as a skeleton of the target story, Peng et al. (2018); Yao et al. (2019) use keywords as a plan of generating stories, Tan et al. (2021) iteratively adds more and more keywords into the plan, Zhong et al. (2022) use the number of extracted events to control the summary granularity, and Mishra and Nouri (2022); Narayan et al. (2023) use question and answer pairs to guide the writing process. Saha et al. (2022); Hosking et al. (2023, 2024); Chowdhury et al. (2024) infer the latent tree structure in the summarization tasks in order to improve its transparency or controllability. Mohri and Hashimoto (2024) use LLM to build a structure similar to our entailment hierarchy to control the trade-off between the factuality and specificity. Chain of thoughts (Wei et al., 2022) or tree of thoughts (Yao et al., 2023b) iteratively generate the thinking process of LLM. Nevertheless, the lack of intermediate supervision often hurts their performance (Lightman et al., 2023; Kabongo et al., 2023).

Our entailment hierarchy is related to entailment graphs (Kotlerman et al., 2015; Hosseini et al., 2018; Cattan et al., 2023). The work discovers the entailment relation between existing sentences while our work generates the summary/story that has the entailment relation with the existing input story/summary. Our iterative story expansion is different from iterative document revision or simplification (Du et al., 2022; Dwivedi-Yu et al., 2022; Schick et al., 2022; Cripwell et al., 2023), where there might not be an entailment relation between the original document and the revised document.

Our proposed entailment hierarchy is also related to entailment tree (Dalvi et al., 2021; Chen et al., 2023) or multi-premise NLI (Lai et al., 2017). However, their hypothesis often needs to be implied by multiple premises together, which is not applicable to the story generation tasks. Similarly, our problem is different from multi-document summarization (Xiao et al., 2022; Wolhandler et al., 2022), multi-section summarization (Stiennon et al., 2020), or sentence union generation (Hirsch et al., 2023). Our goal is to find the intersection statement mentioned in every input while their goal is to have the union statement that covers the important facts in all inputs.

6 Conclusion

In this work, we propose the C2F-StoryTree framework, which allows the user to plan the story plots with the LM. By allowing users to see and participate in the LM's plot planning process, the LM story generation system becomes much more transparent, controllable, and trustworthy (Yang et al., 2022b). For example, the user can select/revise the story summaries to get the desired plot before reading long-generated stories, or let the system automatically select the summary with the best reranker score.

We propose cost-efficient ways to realize the C2F-StoryTree framework using existing story datasets and crowdsourcing. To reduce the cost, we convert the coarse-to-fine story writing tasks into fine-to-coarse summary writing tasks. Moreover, we propose the common summary task to control the specificity of the summary in an entailment hierarchy (EH). By combining several (novel) techniques, we are able to spend less than 1k dollars to construct our EH dataset.

In our experiments, we demonstrate (i) In the

common summary generation task, a T5-3b and reranker fine-tuned on the collected dataset outperform GPT3.5 by a large margin. (ii) The specificities of common summaries let us discover that the SFT and RLHF in GPT3.5 improve the story quality given the general prompts much more than specific prompts. (iii) Our C2F-StoryTree allows the users to steer the story generation toward their desired direction (i.e., higher sBERT similarities with the reference story in our simulation experiment). (iv) The collected dataset drastically improves the short story generation capability of the open-source models. After fine-tuning Vicuna 7B, we achieve much more diverse and creative stories than GPT3.5 given specific prompts.

7 Limitations

The main goal of this paper is NOT to build a state-of-the-art story generation system. Instead, the goal of this paper is to propose novel ways to construct the dataset that can support a coarse-to-fine story generation system and alleviate copyright concerns. Our experiments focus on showing our novel dataset construction method is better than other alternatives and our resulting datasets can improve the story generation ability of LMs. Therefore, we do not compare our methods with the latest LLMs and prompting techniques.

Our methods rely on all-mpnet-base-v2, a state-of-the-art pretrained sentence BERT (sBERT), to estimate the similarities between two stories and similarities between two summaries. Although the off-the-shelf similarity model tends to give a high similarity score to two stories sharing some topics or mentioning similar keywords, we observe that it seems to be insensitive to the differences in story plots with the same topic (Xu et al., 2023). Due to the imperfect similarity measurement, our collected common summaries are usually not very specific, which prevents us from building a deeper tree and degrades C2F-StoryTree's capability of handling more specific prompts/summaries. This also causes difficulties in simulating user selection in our experiments.

The quality of our generated stories is limited by our training data. Most of the professionally written stories are long and protected by copyright and not allowed for commercial usage, so we choose to build our entailment hierarchy on ROC stories, which are crowdsourced short stories and have the CC BY-SA 4.0 license. Nevertheless, the choice limits the upper bound of the quality of our generated stories. Moreover, the quality of our generated long stories still has plenty of room for improvement, especially its coherency (see Appendix F.2). One reason is that the stories in the ROC dataset are very different from the ones in the WP dataset in terms of story themes and writing styles. Another reason is the high cost of manually summarizing a long story into a short story.

Finally, ChatGPT/GPT3.5 could often achieve performance comparable to the fine-tuned state-of-the-art models in non-adversarial NLP tasks (Pikuliak, 2023; Laskar et al., 2023) and could perform much better on summarization tasks (Goyal et al., 2022), so it is surprising that GPT3.5 often fails to generate a common summary and its performance almost remains the same after seeing more training examples or the manual prompt tuning in Appendix D.1, while a T5-3b model with a reranker fine-tuned on only 3.4k labels could do much better. In this study, we did not analyze the main reasons for failure. For example, is GPT3.5's error from its weakness of entailment inference (Kıcıman et al., 2023; Gao et al., 2023; Jin et al.) for stories, its knowledge on generating multi-document summaries, or not understanding the task instruction because it never sees similar tasks before.

8 Ethical and Broader Impact Statement

By fine-tuning LLM using the licensed stories, the LLM would often output the variants of the stories in the fine-tuning dataset. Thus, we can also reduce copyright disputes and establish a mechanism for sharing profits with writers. We can convert our entailment hierarchy into multi-turn dialogues to supervisedly fine-tune LLMs (see Appendix C.2) or evaluate LLMs. We can also generalize our summaries to include more types of prompt constraints such as keywords, which allows us to better control the specificity of prompts and reduce the cost of collecting supervised fine-tuning data for LLM (Li et al., 2023a) through the transitive closure. In this study, we conduct all our experiments on stories, but our entailment hierarchy could be applied to any text genre such as Wikipedia, scientific papers, the latest news, and private technical documents inside a company.

Our proposed common summary task can also be used in not only story/text generation but also fine-grained text categorization or information retrieval. Previously, the taxonomy and ontology

were usually manually defined and usually coarse-grained. We can recursively build the ultra-fine-grained taxonomy using our common summary task and human-and-machine-in-the-loop construction approach.

Finally, the wide applicability of our approach poses a risk. Malicious users could potentially build an entailment hierarchy on biased or toxic stories and generate similar stories with lower costs. Therefore, in each application, proper content filtering techniques should be adopted to mitigate the risk.

References

Max Bartolo, Tristan Thrush, Sebastian Riedel, Pontus Stenetorp, Robin Jia, and Douwe Kiela. 2022. Models in the loop: Aiding crowdworkers with generative annotation assistants. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3754–3767, Seattle, United States. Association for Computational Linguistics. 5

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2022. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*. 7, 15

Arie Cattan, Lilach Eden, Yoav Kantor, and Roy Bar-Haim. 2023. From key points to key point hierarchy: Structured and expressive opinion summarization. *arXiv preprint arXiv:2306.03853*. 8

Kent K Chang, Mackenzie Cramer, Sandeep Soni, and David Bamman. 2023. Speak, memory: An archaeology of books known to chatgpt/gpt-4. *ArXiv preprint*, abs/2305.00118. 2

Ting-Yun Chang and Robin Jia. 2023. Data curation alone can stabilize in-context learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8123–8144. 18

Gang Chen, Yang Liu, Huanbo Luan, Meng Zhang, Qun Liu, and Maosong Sun. 2019. Learning to predict explainable plots for neural story generation. *ArXiv preprint*, abs/1912.02395. 8, 23

Howard Chen, Ramakanth Pasunuru, Jason Weston, and Asli Celikyilmaz. 2023. Walking down the memory maze: Beyond context limit through interactive reading. *arXiv preprint arXiv:2310.05029*. 8

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. 2, 6

Somnath Basu Roy Chowdhury, Nicholas Monath, Avinava Dubey, Manzil Zaheer, Andrew McCallum, Amr Ahmed, and Snigdha Chaturvedi. 2024. Incremental extractive opinion summarization using cover trees. *arXiv preprint arXiv:2401.08047*. 8

Elizabeth Clark, Anne Spencer Ross, Chenhao Tan, Yangfeng Ji, and Noah A Smith. 2018. Creative writing with a machine in the loop: Case studies on slogans and stories. In *23rd International Conference on Intelligent User Interfaces*, pages 329–340. 5

Together Computer. 2023. Redpajama-data: An open source recipe to reproduce llama training dataset. 17

Liam Cripwell, Joël Legrand, and Claire Gardent. 2023. Document-level planning for text simplification. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 993–1006, Dubrovnik, Croatia. Association for Computational Linguistics. 8

Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. 2021. Explaining answers with entailment trees. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7358–7370, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. 8

Mehdi Drissi and Jugal Kalita. 2018. Hierarchical text generation using an outline. In *International Conference on Natural Language Processing*. 8, 23

Wanyu Du, Vipul Raheja, Dhruv Kumar, Zae Myung Kim, Melissa Lopez, and Dongyeop Kang. 2022. Understanding iterative revision from human-written text. In *60th Annual Meeting of the Association for Computational Linguistics, ACL 2022*, pages 3573–3590. Association for Computational Linguistics (ACL). 8

Jane Dwivedi-Yu, Timo Schick, Zhengbao Jiang, Maria Lomeli, Patrick Lewis, Gautier Izacard, Edouard Grave, Sebastian Riedel, and Fabio Petroni. 2022. Editeval: An instruction-based benchmark for text improvements. *arXiv preprint arXiv:2209.13331*. 8

Ronen Eldan and Yuanzhi Li. 2023. Tinystories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*. 2

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics. 4, 20, 21

Le Bronnec Florian, Verine Alexandre, Negrevergne Benjamin, Chevaleyre Yann, and Allauzen Alexandre. 2024. Exploring precision and recall to assess the quality and diversity of llms. 8

Jinglong Gao, Xiao Ding, Bing Qin, and Ting Liu. 2023. Is chatgpt a good causal reasoner? a comprehensive evaluation. *ArXiv preprint*, abs/2305.07375. 9

Xinyang Geng and Hao Liu. 2023. Openllama: An open reproduction of llama. 17

Tanya Goyal, Junyi Jessy Li, and Greg Durrett. 2022. News summarization and evaluation in the era of gpt-3. *ArXiv preprint*, abs/2209.12356. 9

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021a. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing. 5

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021b. Deberta: decoding-enhanced bert with disentangled attention. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. 5

Eran Hirsch, Valentina Pyatkin, Ruben Wolhandler, Avi Caciularu, Asi Shefer, and Ido Dagan. 2023. Revisiting sentence union generation as a testbed for text consolidation. *ArXiv preprint*, abs/2305.15605. 8

Chris Hokamp, Demian Gholipour Ghalandari, Nghia The Pham, and John Glover. 2020. Dyne: Dynamic ensemble decoding for multi-document summarization. *ArXiv preprint*, abs/2006.08748. 5, 6

Tom Hosking, Hao Tang, and Mirella Lapata. 2023. Attributable and scalable opinion summarization. *ArXiv preprint*, abs/2305.11603. 8

Tom Hosking, Hao Tang, and Mirella Lapata. 2024. Hierarchical indexing for retrieval-augmented opinion summarization. *arXiv preprint arXiv:2403.00435*. 8

Mohammad Javad Hosseini, Nathanael Chambers, Siva Reddy, Xavier R. Holt, Shay B. Cohen, Mark Johnson, and Mark Steedman. 2018. Learning typed entailment graphs with global soft constraints. *Transactions of the Association for Computational Linguistics*, 6:703–717. 8

Daphne Ippolito, Florian Tramèr, Milad Nasr, Chiyuan Zhang, Matthew Jagielski, Katherine Lee, Christopher A Choquette-Choo, and Nicholas Carlini. 2022a. Preventing verbatim memorization in language models gives a false sense of privacy. *arXiv preprint arXiv:2210.17546*. 2

Daphne Ippolito, Ann Yuan, Andy Coenen, and Sehmon Burnam. 2022b. Creative writing with an ai-powered writing assistant: Perspectives from professional writers. *arXiv preprint arXiv:2211.05030*. 8

Sophie Jentzsch and Kristian Kersting. 2023. Chatgpt is fun, but it is not funny! humor is still challenging large language models. *arXiv preprint arXiv:2306.04563*. 2

Zhijing Jin, Yuen Chen, Felix Leeb, Ojasv Kamal, Zhiheng Lyu, Kevin Blin, Fernando Gonzalez, Max Kleiman-Weiner, Luigi Gresele, Mrinmaya Sachan, et al. Cladder: Assessing causal reasoning in language models. 9

Salomon Kabongo, Jennifer D'Souza, and Sören Auer. 2023. Zero-shot entailment of leaderboards for empirical ai research. *arXiv preprint arXiv:2303.16835*. 8

Emre Kıcıman, Robert Ness, Amit Sharma, and Chenhao Tan. 2023. Causal reasoning and large language models: Opening a new frontier for causality. *ArXiv preprint*, abs/2305.00050. 9

Lili Kotlerman, Ido Dagan, Bernardo Magnini, and Luisa Bentivogli. 2015. Textual entailment graphs. *Natural Language Engineering*, 21(5):699–724. 8

Wojciech Kryscinski, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. 2022. BOOKSUM: A collection of datasets for long-form narrative summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6536–6558, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics. 20, 21

Alice Lai, Yonatan Bisk, and Julia Hockenmaier. 2017. Natural language inference from multiple premises. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 100–109, Taipei, Taiwan. Asian Federation of Natural Language Processing. 8

Md Tahmid Rahman Laskar, M Saiful Bari, Mizanur Rahman, Md Amran Hossen Bhuiyan, Shafiq Joty, and Jimmy Xiangji Huang. 2023. A systematic study and comprehensive evaluation of chatgpt on benchmark datasets. *ArXiv preprint*, abs/2305.18486. 9

Yukyung Lee, Soonwon Ka, Bokyung Son, Pilsung Kang, and Jaewook Kang. 2024. Navigating the path of writing: Outline-guided text generation with large language models. *arXiv preprint arXiv:2404.13919*. 8

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics. 5, 22

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Luke Zettlemoyer, Omer Levy, Jason Weston, and Mike Lewis. 2023a. Self-alignment with instruction backtranslation. *arXiv preprint arXiv:2308.06259*. 3, 9

Yunzhe Li, Qian Chen, Weixiang Yan, Wen Wang, Qinglin Zhang, and Hari Sundaram. 2023b. Enhancing generation through summarization duality and explicit outline control. *ArXiv preprint*, abs/2305.14459. 8, 21, 23

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *ArXiv preprint*, abs/2305.20050. 8

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics. 6

Alisa Liu, Swabha Swayamdipta, Noah A. Smith, and Yejin Choi. 2022. WANLI: Worker and AI collaboration for natural language inference dataset creation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6826–6847, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics. 5, 20

Yang Liu. 2019. Fine-tune bert for extractive summarization. *ArXiv preprint*, abs/1903.10318. 20, 21

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv preprint*, abs/1907.11692. 24

Sewon Min, Suchin Gururangan, Eric Wallace, Hannaneh Hajishirzi, Noah A Smith, and Luke Zettlemoyer. 2023. Silo language models: Isolating legal risk in a nonparametric datastore. *arXiv preprint arXiv:2308.04430*. 2, 7

Swaroop Mishra and Elnaz Nouri. 2022. Help me think: A simple prompting strategy for non-experts to create customized content with models. *arXiv preprint arXiv:2208.08232*. 1, 8

Christopher Mohri and Tatsunori Hashimoto. 2024. Language models with conformal factuality guarantees. *arXiv preprint arXiv:2402.10978*. 8

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. *ArXiv preprint*, abs/1604.01696. 2, 5, 6

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics. 5

Shashi Narayan, Joshua Maynez, Reinald Kim Amplayo, Kuzman Ganchev, Annie Louis, Fantine Huot, Anders Sandholm, Dipanjan Das, and Mirella Lapata. 2023. Conditional generation with a question-answering blueprint. *Transactions of the Association for Computational Linguistics*, 11:974–996. 8

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901, Online. Association for Computational Linguistics. 20

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744. 7, 17

Nanyun Peng, Marjan Ghazvininejad, Jonathan May, and Kevin Knight. 2018. Towards controllable story generation. In *Proceedings of the First Workshop on Storytelling*, pages 43–49, New Orleans, Louisiana. Association for Computational Linguistics. 1, 8, 22

Matúš Pikuliak. 2023. Chatgpt survey: Performance on nlp datasets. `https://www.opensamizdat.com/posts/chatgpt_survey`. 9

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9. 5

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67. 2, 5

Mathieu Ravaut, Shafiq Joty, and Nancy Chen. 2022. SummaReranker: A multi-task mixture-of-experts re-ranking framework for abstractive summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4504–4524, Dublin, Ireland. Association for Computational Linguistics. 4

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics. 5, 24

Alex Reisner. 2023. Revealed: The authors whose pirated books are powering generative ai. *The Atlantic*. 2, 7

Swarnadeep Saha, Shiyue Zhang, Peter Hase, and Mohit Bansal. 2022. Summarization programs: Interpretable abstractive summarization with neural modular trees. *ArXiv preprint*, abs/2209.10492. 8

Timo Schick, Jane Dwivedi-Yu, Zhengbao Jiang, Fabio Petroni, Patrick Lewis, Gautier Izacard, Qingfei You, Christoforos Nalmpantis, Edouard Grave, and Sebastian Riedel. 2022. Peer: A collaborative language model. *arXiv preprint arXiv:2208.11663*. 8

Abigail See, Aneesh Pappu, Rohun Saxena, Akhila Yerukola, and Christopher D Manning. 2019. Do massively pretrained language models make better storytellers? *arXiv preprint arXiv:1909.10705*. 1

Prasann Singhal, Tanya Goyal, Jiacheng Xu, and Greg Durrett. 2023. A long way to go: Investigating length correlations in rlhf. *arXiv preprint arXiv:2310.03716*. 18

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F. Christiano. 2020. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. 8

Bowen Tan, Zichao Yang, Maruan Al-Shedivat, Eric Xing, and Zhiting Hu. 2021. Progressive generation of long text with pretrained language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4313–4324, Online. Association for Computational Linguistics. 8, 22

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*. 17

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax. 6

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837. 8

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics. 20

Ruben Wolhandler, Arie Cattan, Ori Ernst, and Ido Dagan. 2022. How "multi" is multi-document summarization? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5761–5769, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics. 8

Wen Xiao, Iz Beltagy, Giuseppe Carenini, and Arman Cohan. 2022. PRIMERA: Pyramid-based masked sentence pre-training for multi-document summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5245–5263, Dublin, Ireland. Association for Computational Linguistics. 8

Zhuohan Xie, Trevor Cohn, and Jey Han Lau. 2023. The next chapter: A study of large language models in storytelling. 1

Jingjing Xu, Xuancheng Ren, Yi Zhang, Qi Zeng, Xiaoyan Cai, and Xu Sun. 2018. A skeleton-based model for promoting coherence among sentences in narrative story generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4306–4315. 1, 8, 22

Shicheng Xu, Liang Pang, Jiangnan Li, Mo Yu, Fandong Meng, Huawei Shen, Xueqi Cheng, and Jie Zhou. 2023. Plot retrieval as an assessment of abstract semantic association. *arXiv preprint arXiv:2311.01666*. 9

Kevin Yang, Dan Klein, Nanyun Peng, and Yuandong Tian. 2022a. Doc: Improving long story coherence with detailed outline control. *ArXiv preprint*, abs/2212.10077. 8, 22

Kevin Yang, Yuandong Tian, Nanyun Peng, and Dan Klein. 2022b. Re3: Generating longer stories with recursive reprompting and revision. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4393–4479, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics. 8

Lili Yao, Nanyun Peng, Ralph Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-and-write: Towards better automatic storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7378–7385. 8, 22

Shunyu Yao, Howard Chen, Austin W Hanjie, Runzhe Yang, and Karthik Narasimhan. 2023a. Collie: Systematic construction of constrained text generation tasks. *arXiv preprint arXiv:2307.08689*. 1

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. 2023b. Tree of thoughts: Deliberate problem solving with large language models. *ArXiv preprint*, abs/2305.10601. 8

Wang You, Wenshan Wu, Yaobo Liang, Shaoguang Mao, Chenfei Wu, Maosong Cao, Yuzhe Cai, Yiduo Guo, Yan Xia, Furu Wei, et al. 2023. Eipe-text:

Evaluation-guided iterative plan extraction for long-form narrative text generation. *arXiv preprint arXiv:2310.08185*. 8

Ann Yuan, Andy Coenen, Emily Reif, and Daphne Ippolito. 2022. Wordcraft: story writing with large language models. In *27th International Conference on Intelligent User Interfaces*, pages 841–852. 1

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2020. PEGASUS: pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 11328–11339. PMLR. 24

Ming Zhong, Yang Liu, Suyu Ge, Yuning Mao, Yizhu Jiao, Xingxing Zhang, Yichong Xu, Chenguang Zhu, Michael Zeng, and Jiawei Han. 2022. Unsupervised multi-granularity summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4980–4995, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics. 8

Wenjie Zhong, Jason Naradowsky, Hiroya Takamura, Ichiro Kobayashi, and Yusuke Miyao. 2023. Fiction-writing mode: An effective control for human-machine collaborative writing. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1752–1765, Dubrovnik, Croatia. Association for Computational Linguistics. 1, 8

A Appendix Overview

In the appendix, we will list our main contributions in Appendix B, provide additional motivations in Appendix C, provide more experiment results in Appendix D, describe our general summary/hypothesis generation method in Appendix E, describe our long story generation method and evaluation in Appendix F, provide a formal definition of entailment hierarchy in Appendix G, describe some experiment details in Appendix H, present the details of our models in Appendix I, and report the details of our crowdsourcing tasks in Appendix J.

B Main Contributions

- We propose C2F-StoryTree, a coarse-to-fine tree-based story generation framework. Compared to the typical story generation workflow, our framework is more controllable and transparent and can lead to high-quality stories that align well with the user's interest.
- We propose a cost-efficient bottom-up way to build an 'entailment hierarchy', a text data structure for realizing the framework, using human-and-machine in the loop and existing public datasets in the creative writing domain.
- We propose a new common summary writing task, which avoids the resulting summary/prompt from being too specific or general. In the task, we can also measure the specificity of the prompt using the similarity of the two input stories, which allows us to test the performance of different models given different prompt specificities.
- We propose a novel method to generate a common summary of two stories using joint decoding and reranker. We demonstrate that our fine-tuned model is much better than GPT3.5 few-shot in-context learning.
- We design a series of MTurk tasks to minimize the cost and maintain the high annotation quality. Our method allows us to build a high-quality 2-layer entailment hierarchy on top of 5-sentence stories by spending less than $1000 on MTurk.
- We design and conduct comprehensive evaluations to compare the quality of common summaries and stories from different generation systems in various aspects.

C Additional Motivations

We motivate our C2F-StoryTree framework and entailment hierarchy datasets from two more perspectives: search and supervised fine-tuning.

C.1 The Story Search Perspective of C2F-StoryTree

When a user interacts with a creative writing LM, his/her goal could be viewed as searching the good stories of interest among all the possible stories generated by the LM, while using a prompt as a constraint to exclude the undesired stories. In Figure 6, we compare the two workflows. The typical workflow randomly picks a story from the set of stories that satisfy the constraint and the user needs to verify each random story sample. The process is inefficient because writing a prompt (constraint construction), random sampling (generating the story), and verification (reading the story) are all very expensive and time-consuming. Motivated by this need, we propose to search in a coarse-to-fine tree-based fashion. Given a prompt, the LM provides some short summaries representing the clusters of stories that satisfy the constraints and these summaries could directly become the prompt in the next round, which greatly accelerates the search process.

C.2 Multi-turn Supervised Fine-Tuning

The impressive story generation capability of LLM relies on its huge training corpus from the Internet (Carlini et al., 2022), but the Internet definitely does not contain all possible good stories or prompts humans could write, especially if we only consider the ones without copyright protection. Therefore, how to cost-efficiently construct the dataset with the prompts/stories LLMs haven't seen before remains a very important research topic in the era of LLM. We can use the dataset to not only evaluate or improve the single-turn story generation ability of LLMs as shown in our experiment, but also easily construct a multi-turn supervised fine-tuning (SFT) dataset.

For example, we can construct SFT that allows the LLM to proactively suggest more specific prompts to help users clarify their intentions as in Figure 1. Besides, we can construct a SFT dataset to help another common scenario where the user is not satisfied with the current response of the LLM and wants to see the alternatives. The following template is an example to reduce the chance of

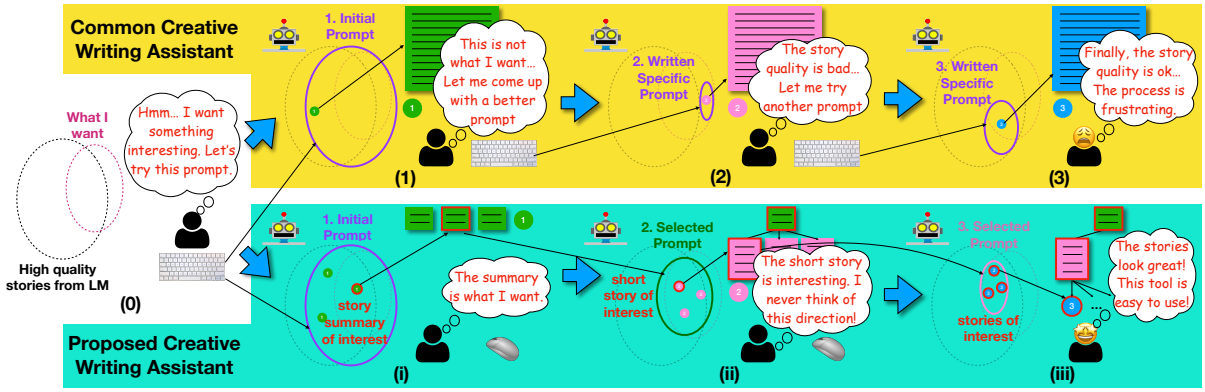


Figure 6: The comparison of existing assistants and our assistant. Each circle refers to a set of stories that satisfy the constraint. (0) a user provides a general initial prompt. **Existing workflow**: (1) The user reads the story and rewrites the prompt. (2) The LM cannot generate a good story for a new specific prompt. (3) The user feels frustrated after reading two long stories and writing two specific prompts. **Proposed workflow**: (i) The user chooses one of the generated specific summaries he/she likes. (ii) The system continues partitioning the story space of the user's interest using three short stories. (iii) Finally, the user loves the three stories after reading some short texts and a few clicks.

generating repeated or too similar responses from LLMs.

Template C.1 ***Human****: Please generate a five-sentence daily-life story containing the following plot:* ***{Summary Input}***
Assistant*: Yes! Here's your story!* ***"{Story A}"***
Human*: Please write another story that is very different from the previous one.*
Assistant*: Of course. Here's a different story for you:* ***"{Story B}"***

D Additional Experimental Results

This paper builds a hierarchical story generation system and we conduct many experiments and analyses to verify our conclusions and contributions. Due to the space constraint in the main paper, we move some results to this section.

In Appendix D.1, we try to use prompt engineering to improve the GPT3.5's results in common story generation. In Appendix D.2, we compare more short stories generated by LLMs using different methods. In Appendix D.3, we provide more explanation of how we analyze generation methods using the prompt specificities and more insights we can get from the analyses. In Appendix D.4, we plot the relation between common summary performances and the training data sizes. In Appendix D.6, we visualize another hierarchical generation result.

D.1 Instructing GPT3.5 to Output More General Common Summaries

Setup: In Table 2, we observe that the outputs of GPT3.5 baselines tend to be too specific to be common summaries. We tried to adjust the prompts to make the baseline outputs more general. We conduct another round of human evaluation for the GPT3.5 baselines and our two best methods and report the results in Table 4.

Methods: According to the definition of the common summary, we originally put *Please be as specific as possible.* into the prompt of **GPT3.5 5 Shot** in Table 2, so we call this baseline **GPT3.5 5 Shot (specific)**. First, we delete the *Please be as specific as possible.* and call this method **GPT3.5 5 Shot (not mention)**. Next, we even insert *Please be as general as possible.* into the prompt and call this method **GPT3.5 5 Shot (general)**.

Results: Although instructing GPT3.5 to output more general common summaries slightly reduces the average length, the specificity does not decrease and the entailment successful rates decrease in the human experiment, which shows the difficulty of controlling the specificity of the generated common summaries through the prompt engineering. In contrast, our proposed method can easily adjust the specificity of the generated common summaries by controlling the ratio between the specific summary training set and the general hypothesis training set.

Method	Model Size	Training Data Size: Specific Summary	Training Data Size: General Hypothesis	Automatic Metric: len	Automatic Metric: Fluency ppl (↓)	Automatic Metric: Diversity (%) dist-1	Automatic Metric: Diversity (%) dist-2	Human Judgement: Entail 2 (%)	Human Judgement: Entail 1 (%)	Human Judgement: Fluency	Human Judgement: Specificity (↓) (%)
GPT3.5 5 Shot (specific)		5		15.87	**9.05**	**37.98**	**69.17**	43	68.25	4.71	20.5
GPT3.5 5 Shot (not mention)	175B	5		14.59	9.69	35.90	63.55	35	63	4.74	19.5
GPT3.5 5 Shot (general)		5		14.17	9.77	36.81	63.55	40.5	65	4.69	**17**
JDC + rerank	3B	3.4k		10.64	10.24	33.18	54.17	54.5	69.75	4.77	37.5
JDC + rerank + hypo		3.4k	1.9k	10.04	10.30	34.29	54.89	**69.5**	**80**	**4.85**	47

Table 4: Comparison of the different prompts of GPT3.5. GPT3.5 5 Shot (specific) is the same as GPT3.5 5 Shot in Table 2. Our two JDC (Joint decoding with concatenation) methods are the same as the ones in Table 2. The human judgment numbers are different compared to Table 2 because they are labeled by a different set of MTurk workers. ↓ means lower is better and the best scores are highlighted. GPT3.5 means *text-davinci-003*.

Method	Story Generator	C2F	Automatic Metric: len	Reference Relevancy (%): R1	R2	sim	Creativity (%): R1 (↓)	R2 (↓)	sim (↓)	Coherence ppl (↓)	Diversity (%): dist-1	dist-2	Human Judgement: Rel	Pro (%)	Cr	Coh	Eng
Round for Comparing with LLMs																	
EH + rerank	T5-3b FT	V	47.70	**25.07**	**3.77**	**46.64**	**16.79**	**3.98**	48.61	9.97	40.63	78.66	**2.49**	**84.17**	**3.14**	**3.86**	**2.85**
CG	OpenLLaMA 5 Shot		48.72	21.67	2.49	40.70	18.56	6.16	**45.01**	**9.92**	**45.57**	**84.46**	2.23	70.00	3.12	3.56	2.58
CG	Vicuna 5 Shot		46.21	23.77	3.21	43.79	20.68	6.98	50.74	10.01	42.54	80.47	2.52	89.17	3.11	4.21	2.79
CG	GPT3.5 1 Shot		64.53	24.21	3.14	45.77	**16.67**	**4.84**	53.57	**9.32**	39.67	**78.54**	**2.92**	**96.67**	**3.58**	4.42	3.42
	GPT3.5 5 Shot		57.91	**24.94**	**3.46**	**47.26**	18.49	5.37	54.41	9.57	**39.71**	78.26	2.90	**96.67**	3.54	**4.51**	**3.44**
ROC Stories	Human		50.49	22.55	2.70	44.99	15.43	3.06	41.84	9.89	-	-	2.16	70.00	3.52	4.14	3.53
Round for Generators using Proprietary Data																	
EH + rerank	T5-3b FT	V	47.70	**25.07**	**3.77**	**46.64**	16.79	3.98	48.61	9.97	40.63	78.66	2.55	76.67	3.22	3.68	3.29
EH	Vicuna 5 Shot	V	49.12	23.08	2.74	42.26	15.89	**3.28**	**45.67**	9.88	**43.90**	**83.80**	2.53	60.83	3.13	3.97	3.07
EH + rerank	Vicuna 5 Shot	V	49.85	24.67	3.56	45.65	17.74	4.16	50.59	9.86	41.43	80.83	2.59	86.67	3.28	**4.16**	**3.38**
EH + rerank	Vicuna FT	V	52.33	24.43	3.68	45.72	**15.56**	3.32	46.39	**9.79**	42.37	81.54	**2.72**	80.00	**3.56**	3.98	3.36
CG	Vicuna 5 Shot		46.21	23.77	3.21	43.79	20.68	6.98	50.74	10.01	42.54	80.47	2.55	**88.33**	3.10	3.99	3.06
CG	GPT3 5 Shot		58.37	**25.15**	**3.46**	**47.57**	18.43	**5.50**	**54.82**	**9.55**	38.93	76.97	2.95	**90.00**	3.42	4.13	3.59
	GPT3.5 5 Shot		57.91	24.94	**3.46**	47.26	**18.49**	5.37	54.41	9.57	**39.71**	**78.26**	**3.02**	**90.00**	**3.67**	**4.35**	**3.63**
Human	ROC NN Stories		50.87	27.75	4.50	61.56	11.87	1.42	36.95	9.86	46.38	86.47	2.78	54.17	3.52	4.20	3.58

Table 5: Comparison of the short stories generated by LLMs. Vicuna means *Vicuna 7B v1.3*, OpenLLaMA means *OpenLLaMA 7B v2*, GPT3.5 means *text-davinci-003* (175B), and GPT3 means *davinci* (175B). CG (GPT3.5 5 Shot), CG (GPT3 5 Shot), CG (Vicuna 5 Shot), EH + rerank (T5-3b FT), and EH + rerank (Vicuna 5 Shot) are the same as the ones in Table 3. Different rounds of human evaluation are done by different sets of MTurk workers, so the scores of the same method might be different in different rounds or in Table 2, and the human scores in different rounds are not directly comparable. ↓ means lower is better and the best scores within each section are highlighted.

D.2 Short Story Generation using LLMs

Setup: We conduct two rounds of human experiments to see whether the collected dataset could be used to improve LLMs. In Table 3, we only present partial results in one of the rounds due to the space constraint. In this subsection, we present more results and analyses.

Methods: In addition to GPT3, GPT3.5, and Vicuna 7B v1.3, we also test the story quality from OpenLLaMA 7B v2 (Touvron et al., 2023; Computer, 2023; Geng and Liu, 2023)[2], and from humans. All the 5-shot methods use the same prompt and all the fine-tuned methods use the same training data.

The **ROC stories** baseline comes from using story A as the reference and story B as the prediction or vice versa. The sampling method of story A and story B could be found in Appendix H.2.1.

Results of Generation Models: We report the results in Table 5. In the LLM evaluation round, our **T5-3b FT** is significantly better than **OpenLLaMA 5 Shot** in almost all the metrics. **Vicuna 5 Shot** is much better than its base model, LLaMA, after being trained on the ShareGPT.com data. This shows the generated story quality heavily depends on the training data quality.

EH (Vicuna 5 Shot) and **CG (Vicuna 5 Shot)** perform almost the same in all the human evaluation metrics except for the instruction following probability. This indicates that coarse-to-fine generation increases the controllability and transparency without decreasing its story quality. On most metrics, the significantly better scores from **EH + rerank (Vicuna 5 Shot)** demonstrate the benefits of the reranker/scorer on LLM (Ouyang et al., 2022), which is a much smaller LM trained by our collected dataset.

Notice that our prompt instructs all LLMs to generate a story with around 50 words. On average, GPT3 and GPT3.5 do not follow this constraint closely and generate longer stories than other ap-

[2] https://huggingface.co/openlm-research/open_llama_7b_v2

proaches. A longer story can use more words to develop characters or introduce more characters. Its additional words can create twists or make the transition in the twists more smooth. Thus, the longer stories might give GPT3 and GPT3.5 some advantages (Singhal et al., 2023), especially on coherency and engagement.

Finally, **CG (GPT3.5 5 Shot)** could achieve better coherence, similar engagement, and similar creativity compared to **CG (GPT3.5 1 Shot)** using shorter story length. This shows that our collected story and summary pairs could also be potentially used to improve GPT3.5 175B performance in the future through example selections for few-shot learning (Chang and Jia, 2023) or fine-tuning.

Results of Human-written Stories: In Table 5, **ROC NN stories** have a much higher sBERT score and significantly higher ROUGE 1 and ROUGE 2 than **EH + rerank (Vicuna FT)**, but its relevancy score is only slightly higher. The result supports our observation of the sBERT limitation on short stories. Similarly, we observe that perplexity does not correlate with the human coherence scores well, so it is no longer a good coherence measurement for evaluating LLMs

An author of this paper writes the common summary of the story A and B, so almost all the stories in **ROC stories** should follow the instruction. The only 70% instruction following rate of **ROC stories** suggests that the crowd workers sometimes judge the instruction following based on the lexical matching and humans sometimes disagree with each other on the entailment judgment, especially when the story A and B are not similar with each other (see more details in Appendix J.4).

D.3 Scores under Different Specificities

Setup: Given a metric, each prompt has a specificity measurement and a score for each method. However, it is hard to compare multiple methods using a scatter plot, so we compare their linear regression trendlines instead. We use the default hyperparameters in Sklearn[3] and plot the resulting lines between the smallest and largest sBERT similarity values (specificities).

Results: In Figure 7 and Figure 8, **EH + rerank + sim (T5-3b FT)** significantly improves the goal relevancy of **EH + rerank (T5-3b FT)** only when the prompt is general. We suspect that the worse goal relevancy for specific prompts is due to the reduced influence of our effective reranker. The finding again confirms that sBERT cannot measure the plot similarity between the two stories sharing the similar topics. Compared to **EH (T5-3b FT)**, **EH + rerank (T5-3b FT)**'s much better instruction relevancy (instruction following probability) shows that our reranker could effectively reduce the negative effect of fine-tuning on the noisy generated summaries in Section 3.

In Figure 9, we observe that **CG (GPT3.5 5 Shot)** has a better coherence and instruction following probability than **CG (GPT3.5 1 Shot)** for specific prompt, which suggests that the additional examples could help GPT3.5 to handle more complex prompt.

In Figure 10, we can see that the **EH + rerank (Vicuna FT)** does better on creativity and goal relevancy but worse on coherence and engagement for specific prompts than **EH + rerank (Vicuna 5 Shot)** because the stories in the ROC dataset have higher diversity but lower quality than the stories in the pretraining dataset. Besides, **EH (Vicuna 5 Shot)** achieves better coherence for general prompt than **CG (Vicuna 5 Shot)** probably because C2F generation could prevent the input prompt from being too general to be in-domain testing data.

Notice that different rounds of evaluation are scored by a different set of annotators and they compare a different set of methods, so the trends of the same method in different rounds might be different. For example, **EH + rerank (T5-3b)**'s coherence decreases as specificities of prompts increase in the open-source round, but the trend is reversed in the LLM round and the restricted model round.

D.4 Common Summary Success Rate Versus Training Dataset Size

While collecting common summaries, we can use the verification data from MTurk to track the growth of success rate (Entail 2) with the increase of training data size. Each round of verification is done by different sets of workers and humans sometimes disagree with each other on the entailment judgments (see Appendix J.4), the success rate of the same generated common summaries could change at different rounds. Therefore, we

[3] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

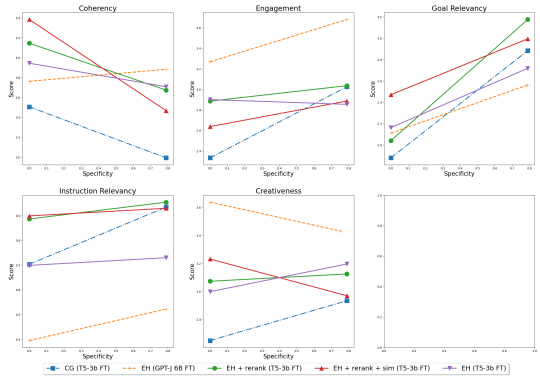


Figure 7: Short story evaluation for open-source round under different prompt specificities

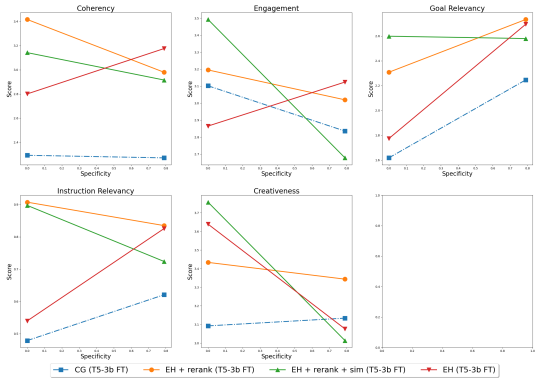


Figure 8: Long story evaluation under different prompt specificities

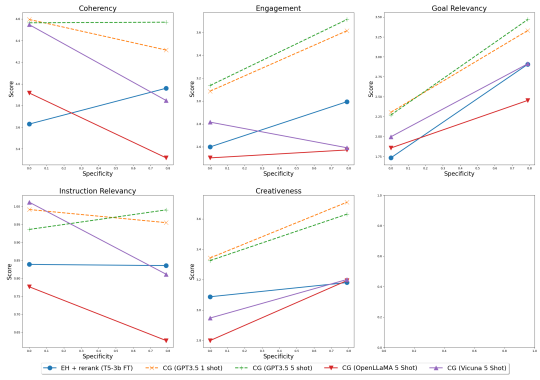


Figure 9: Short story evaluation for LLM round under different prompt specificities

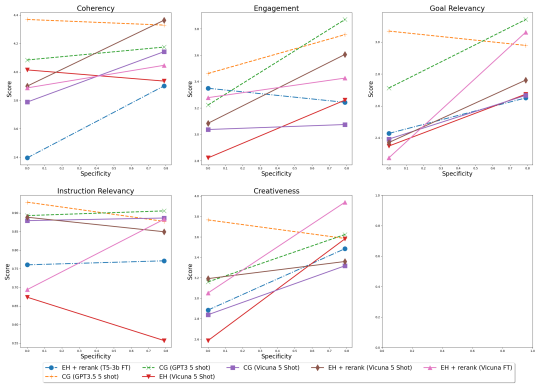


Figure 10: Short story evaluation for restricted model round under different prompt specificities

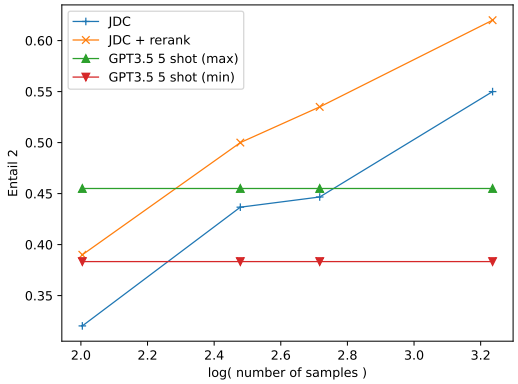


Figure 11: The success rate of generated common summaries versus the log of training data size.

compare our method with the maximum and minimum success rate of GPT3.5 5 shot in Figure 11. We can see that our success rate increases roughly linearly with the log of sample numbers and surpasses the best GPT3.5 5 shot performance significantly with less than 1k samples.

Method	ROUGE1	ROUGE2	sBERT sim
C	18.84	3.60	41.15
C + rerank	19.04	3.21	44.08
JD	19.26	3.25	44.20
JD + rerank	20.16	4.49	50.95
JDC	20.11	3.24	44.97
JDC + rerank	22.38	4.54	51.52

Table 6: Comparison of different common summary generation methods in low-resource setting (with only 100 samples).

D.5 Comparison with only Inputting the Story Concatenation

To support our claim in Section 2.2.1, we compare our methods with the model that uses only the concatenation (**C**) as input. First, we compare all the methods using the similarity between the generated common summary and the reference common summary after we collected the first batch of 100 samples. The testing data come from our short story generation experiment.

We can see from Table 6 that when the training dataset is small, **JD** (input story A and story

B separately) is significantly better than **C**, and **JDC** is significantly better than **JD**. In this paper, the main purpose of predicting the common summary is to reduce the cost of human labeling, so its performance under low-resource is important.

After we have thousands of training examples in Table 2, we found that the Entail 2 score of **C** is 0.325, which is very close to the 0.33 from **JDC**, which shows that **JDC** does similar to **C** when we have more examples. In sum, **JD** does well when we just start to collect the training samples. On the other hand, **C** does well when we have collected lots of training samples. Our proposed **JDC** could do well in both low-resource and high-resource settings.

D.6 Another Visual Example

Table 7 visualizes the prompts generated by our methods to see how well our method could handle a more specific input prompt. We can see that our method can generate diverse specific summaries so that the user can choose the most relevant one as the prompt for generating the short stories. **CG** still has the undesired tendency of copying too many words from the input prompt.

E Construction of the General Summary / Hypothesis Layer

When applying the common summary generation method in Section 2 to build the top layer, we encounter one major challenge: we notice that given two specific summaries, workers sometimes are not able to write a common hypothesis that is significantly more general than both specific summaries (e.g., *A man needed lots of money* and *People won lots of money*). To solve this challenge, we allow the workers to select one appropriate specific summary (usually the most specific one) from 5 generated candidates (see Appendix J.3 for more details).

When we try to automatically generate the hypotheses, we don't know which pair of specific summaries humans would choose, so we can only input one single specific summary and output its general hypothesis candidates. Then, we ask humans to verify the entailment relation between the input specific summary and generated candidates. After collecting sufficient entailment pairs, we train T5-3b^{h2s} model in Figure 2, which generates the specific summaries given the general hypothesis.

As in Section 2.3, we use sBERT to find similar specific summaries and collect their common general hypotheses using MTurk. We use the entailment pairs in three NLI datasets, MNLI (Williams et al., 2018), ANLI (Nie et al., 2020), and WANLI (Liu et al., 2022) to pretrain a T5-3b model and a DeBERTaV3-large reranker. Although the datasets cover various domains, we still collect the common hypotheses of story summaries for fine-tuning our LM to prevent generating paraphrases. In total, we spend 300 US dollars to collect 924 entailment pairs among 1921 labels between 1170 specific summaries and 1042 general hypotheses.

F Long Story Generation

In this study, we spend most of our MTurk budgets on short story generation dataset construction and evaluation because (i) Reading the long story and summarizing it into a short story is very expensive. (ii) It is very hard to generate a coherent long story based only on open-source models and crowd-written long story datasets.

Nevertheless, as illustrated in Figure 2, we still leverage the existing datasets to train our T5-3b model. In this section, we compare the different generation methods to test our framework for long story generation.

F.1 Method for Generating a Long Story from a Short Story

There are no public summarization datasets whose summary is a short story (to the best of our knowledge), while it would be very expensive to ask MTurk workers to summarize a long story into a 5-sentence story. Hence, we tried two methods to overcome this challenge. First, BookSum (Kryscinski et al., 2022) provides long summaries of book chapters and creates a paragraph-level summarization dataset by mapping each sentence in a chapter summary to its most similar paragraph in the book chapter. We modify the process by viewing 5 consecutive sentences in a chapter summary as a short story and map it to the book chapter paragraph that is most similar to the short story. Using the modified BookSum dataset, we can train an abstract summarization T5-3b model. In the second approach, we use the prompt in WritingPrompts (WP) (Fan et al., 2018) as a noisy summary and train an extractive summarizer based on RoBERTa-base[4] (Liu, 2019). During testing, we extract 5

[4] https://github.com/HHousen/TransformerSum

General Summary/Hypothesis as the Input Prompt: "A group of people found a threat in the woods and the enemy ran away."	
Reference Short Story: "The couple went to take a hike in the woods. They hiked for miles. On the way the encountered a big ferocious bear. The man screamed at the bear. The bear ran away in terror. "	
Short Story from **CG** (Conditional Generation without Entailment Hierarchy): "A group of people were out for a walk one night. They came across a threat in the woods. The enemy attacked and the people ran from the danger. The enemy was never seen again!"	
Stories from **EH + rerank + sim** (T5-3b FT)	
Specific Summary from T5-3b^{h2s}	Short Story from T5-3b^{s2s}
1. A group of people were out in the woods when they came across a threat. The group rushed to defend themselves and their property and the intruder had to retreat quickly.	
2. A family was out camping when they came across a stray dog. The dog was scared and ran off into the woods.	1. Sam and his friends went to the woods for the first time. They were afraid a bear would come out of the woods. The bear ran away so the kids had to hide. The kids found a large twig and snuck up to it. The twig protected the children from the bear
3. A group of people were in the woods when they encountered an enemy. They were able to successfully engage the enemy in a fight, but the enemy was able to get away from them.	
4. A group of people go hunting in the woods. They find a wild animal and try to catch it. The creature runs off and the group is successful.	2. The boys were walking through the forest when they saw an animal. They stopped to watch as the animal ran in the other direction. The boys ran towards the nearest campsite for cover. They were afraid and had to find a way to escape. The boys found some sticks to throw in the water to scare it away.
5. A group of people were out in the woods when they came across an unknown person and the person fled the scene.	
6. A group of people were hunting in the woods when they encountered an enemy and the enemy ran away.	

Table 7: Example summaries and stories generated by C2F-StoryTree. We highlight the text that is most similar to the reference/goal story.

				Automatic Metric									Human Judgement				
				Reference Relevancy (%)			Creativity (%)			Coherence	Diversity (%)		Rel	Pro	Cr	Coh	Eng
Method	Story Generator	C2F	len	R1	R2	sim	R1 (↓)	R2 (↓)	sim (↓)	ppl (↓)	dist-1	dist-2		(%)			
Long Story for Generators with Open-Source Licenses																	
CG	T5-3b FT		423.97	11.73	1.69	34.66	**5.37**	2.15	40.11	**3.37**	17.25	53.90	1.97	55.83	3.12	2.28	2.95
EH		V	306.34	14.36	2.07	37.72	5.39	**0.99**	**36.28**	4.34	20.89	58.53	2.29	70.00	3.31	3.01	3.01
EH + rerank		V	279.79	15.22	2.13	39.59	5.90	1.29	40.00	4.30	**21.77**	**60.42**	2.55	**86.67**	**3.38**	**3.17**	**3.10**
EH + rerank + sim		V	288.52	**16.25**	**2.62**	**47.38**	5.78	1.18	40.02	4.14	20.29	57.36	**2.59**	80.00	3.35	3.01	3.04

Table 8: Comparison of the generated long stories. The meaning of symbols are the same as the ones in Table 3.

sentences and arrange them into a noisy short story using their original order in the long story.

In a preliminary manual evaluation, the first abstract summarization method results in better summary quality, and the second extractive summarization method could encourage the story generator to follow the plot of the short story more closely by copying some sentences from the short story. Therefore, we use the first approach to summarize the stories in the training set of WP and the second approach to summarize its validation set. The resulting entailment pairs from both methods become the training data of our T5-3b^{s2l} long story generator.

We leverage the existing datasets to collect the entailment pair of a short story and a long story. We use BookSum (Kryscinski et al., 2022) and T5-3b to train an abstract summarizer. Then, we use WritingPrompts (WP) (Fan et al., 2018) and BERTSUM (Liu, 2019) to train an extractive summarizer. We generate the summaries of WP and train T5-3b^{s2l} to generate long stories given the summaries in Figure 2.

F.2 Long Story Generation Experiment

Setup: In the long story generation experiments, we still utilize the 60 prompts and 120 reference stories created for evaluating the short story models. When computing the reference relevancy, we compare the similarity between the reference short story and the generated long stories.

Methods: The condition generation **CG** baseline first trains an abstract summarizer using the BookSum paragraph dataset, generates the one-sentence summary of every WP story, and trains a T5-3b model to generate the long stories given the summary (Li et al., 2023b).

The other methods except **EH + rerank + sim** just convert the generated short stories to long stories using our long story generator T5-3b^{s2l}. To simulate the user's selection of the short stories, **EH + rerank + sim** selects the short stories using both reranker and sBERT in the second iteration instead of only using reranker in the short story generation experiments. Please see Table 9 for more details.

Results: In Table 8, we can see that the long story quality improvement of **EH + rerank** over **CG** is

significantly larger than the short story quality improvement in Table 3 in terms of diversity, prompt relevancy (i.e., instruction following probability), and coherency. The results show that ROC stories and our entailment hierarchy dataset successfully scaffold the long story generation process and support the findings of previous course-to-fine generation studies such as Xu et al. (2018); Peng et al. (2018); Yao et al. (2019); Tan et al. (2021); Yang et al. (2022a).

G Formal Definition of Entailment Hierarchy

We assume we have an entailment hierarchy (EH) with height I. Let's denote the jth text node at ith layer as T^i_j. The top layer is 1st layer and the bottom layer is Ith layer. If there is an edge between $T^i_{j_p}$ and $T^{i+1}_{j_c}$, $T^{i+1}_{j_c}$ entails $T^i_{j_p}$. Notice that we do not know if the two nodes without an edge have the entailment relation or not. For $i = 2 \dots I$, each T^i_j has only one parent node. If T^{i+1}_A and T^{i+1}_B both have an edge to T^i_j, we said T^i_j is an common summary of T^{i+1}_A and T^{i+1}_B.

After we have built entailment hierarchy dataset, we can train a seq2seq model to map from each parent node $T^i_{j_p}$ to its child nodes $T^{i+1}_{j_c}$. When a user provides an unseen prompt T^i_j, we can generate its M possible child nodes T^{i+1}_{N+1} ... T^{i+1}_{N+M} for the user to choose. Assuming the user can choose T^{i+1}_{N+1} as the new prompt, and we generate the child nodes of T^{i+1}_{N+1} in the next iteration. The process would stop until the user reach the Ith layer at the bottom.

H Experiment Details

We set k=10 in top k sampling to reduce the chance of generating invalid summaries or the story that does not respect the constraint of the prompt. We use Dist-n (Li et al., 2016) to measure the diversity of stories. In all the generated stories, Dist-n is the number of unique n-grams divided by the number of words. We use a NLTK word tokenizer[5] to measure the story length and to get the n-gram for Dist-n. We lowercase the words before computing Dist-n. Similarly, we use stemmer to reduce the mismatch of similar words when measuring the ROUGE F1 scores. We measure the fluency using the perplexity (ppl) of a GPT-2 XL model and similarity using all-mpnet-base-v2.

[5] https://www.nltk.org/api/nltk.tokenize.html

Some MTurk workers tend to give high scores to all the stories and others might prefer to give lower scores. To reduce the variance of the scores, we ask the workers to compare all methods in each question and encourage them not to give the same score to every method. Furthermore, every human judgment is made by two MTurk workers and we report the average of their scores.

H.1 Common Summary Generation

For each input story pairs, each method generates 10 different common summaries in order to compute their dist-n diversity metrics. Since the entailment relation has the transitivity property, we can estimate the entailment between the story and the general summary/hypothesis by traversing the tree and removing the duplicate pairs. The transitivity closure brings us 792 additional entailment pairs to boost our LM's capability in summarizing the single story in Table 2 and thus, increases the performance of generating the common summary through our joint decoding method.

We provide a screenshot of our instruction and evaluation task in Figure 13. For measuring the specificity in human evaluation, we select the additional story C whose common summary is similar to the common summary of story A and story B (e.g., a cousin node in the tree). When the two workers disagree on an entailment relation (one person believes the pair has an entailment relation but the other does not), we label the pair of generated summary and the story as positive. In Table 2, the average disagreement probability for stories A, B, C are 19%, 22%, and 19%, respectively. The average Pearson correlation for fluency is 39%. During our evaluation, 94% of human judgment is provided by 6 different workers.

H.2 Story Generation

Evaluating the one-shot story generation systems is complicated let alone our coarse-to-fine framework. In this section, we describe the detailed design choices we made in our experiment.

H.2.1 Setup Details

To let our testing user prompts have different specificities, we first sample one story A and find its nearest neighbor in the sBERT embedding space. Besides this story pairs with the highest similarity, we find other three story B whose similarities to story A is this highest similarity minus 0.1, minus 0.25 (for a not-so-similar story B), and minus 0.5

	Iteration 1 Hypothesis to Summary (h2s)	Iteration 2 Summary to Short Stories (s2s)	Iteration 3 Short Story to Long Stories (s2l)
Round for Short Story Generators with Open-Source Licenses			
CG	T5-3b$^{cg\ short}$		-
EH	T5-3b^{h2s}	T5-3b^{s2s}	-
EH + rerank	T5-3b^{h2s} + Reranker	T5-3b^{s2s} + Reranker	-
EH + rerank + sim	T5-3b^{h2s} + Reranker + Sim	T5-3b^{s2s} + Reranker	-
EH (GPT-J 6B)	T5-3b^{h2s}	GPT-J 6B	-
Round for Short Story Generators using Proprietary Data			
CG	Vicuna 5 Shot / GPT3.5 5 Shot / GPT3 5 Shot		-
EH (Vicuna 5 Shot)	T5-3b^{h2s}	Vicuna 5 Shot	-
EH + rerank (Vicuna 5 Shot)	T5-3b^{h2s} + Reranker	Vicuna 5 Shot + Reranker	-
EH + rerank (Vicuna FT)	T5-3b^{h2s} + Reranker	Vicuna FT + Reranker	-
Round for Long Story Generators			
CG	T5-3b$^{cg\ long}$		
EH	T5-3b^{h2s}	T5-3b^{s2s}	T5-3b^{s2l}
EH + rerank	T5-3b^{h2s} + Reranker	T5-3b^{s2s} + Reranker	T5-3b^{s2l}
EH + rerank + sim	T5-3b^{h2s} + Reranker + Sim	T5-3b^{s2s} + Reranker + Sim	T5-3b^{s2l}

Table 9: The models we use to generate the stories at each iteration. Sim means selecting the prompt that is most similar to the reference story. Notice that we do not use Sim at the last (rightmost) iteration to make the similarity comparison of the generated story fair.

(for a dissimilar story B). Using the sampling methods, we prepare 60 testing story pairs. After writing the common summaries for them, we can get 120 testing pairs of summary and reference story (since each common summary corresponds to both story A and B).

In our automatic evaluation, we use the generated 3 stories for each testing pair to compute their generation diversity and other automatic metrics. In our human experiments, we select one of the generated stories for each testing pair, and each generated story receives 2 scores from different MTurk workers for each metric, so each human evaluation score is the average of 240 scores from humans.

One metric of our human evaluation is the creativity of incorporating the prompt into the output story and directly copying a large portion of the text from the prompt would degrade such creativity. Hence, we do not ask workers to estimate the creativity of the output story if it does not follow the prompt instruction.

Compared with the short stories we generated, the generated long stories have more coherence issues and are less similar to the reference story. To prevent the workers from giving universally low scores, we adjust the scoring guideline to encourage the workers to focus more on the main plots of long stories when judging the coherency and relevancy.

We provide the screenshot for short story evaluation in Figure 14a and that for long story evaluation in Figure 14b. In each task, we randomize the order of stories from different systems to prevent positional bias. In the open-source round of Table 3, the average Pearson correlation between the scores from two MTurk workers is 40% for goal/reference relevancy, 46% for prompt following, 46% for coherency, and 23% for engagement/interestingness. Finally, 78% of the tasks are done by 8 MTurk workers.

H.2.2 Method Details

At each iteration during C2F generation, we test three methods to select the prompts/summaries/stories. **EH** chooses 3 prompts randomly; **EH + rerank** chooses 3 out of 30 prompts with the highest reranker score; **EH + rerank + sim** first chooses 10 out of 30 prompts using the reranker, and simulates user's interaction by choosing the summary that has the highest text similarity (measured by all-mpnet-base-v2) with the reference story.

T5-3b is not fine-tuned for story generation, so we need to do additional fine-tuning without using our collected entailment hierarchy for the **CG** baseline in the T5-3b round. Thus, we adopt the approach in Drissi and Kalita (2018); Chen et al. (2019); Li et al. (2023b): first, we train a summarization model using the BookSum paragraph dataset. Then, we generate a one-sentence summary of all ROC stories and train a T5-3b model to generate the short stories given the summary.

In Table 9, we report the models each method

uses at each iteration. T5-3b$^{cg\ short}$ and T5-3b$^{cg\ long}$ refer to the conditional generator trained to output ROC stories and WP stories, respectively. Specifically, **CG** and its T5-3b$^{cg\ long}$ are trained using the training and validation set of WP as **EH + rerank** did. When fine-tuning **EH (GPT-J 6B)**, we prepend "Given a story summary:" to the prompt and "Please write the original story" to the story. The maximal length of long stories for both T5-3b and GPT-J 6B is 1024.

H.3 LLM Baselines

To reduce the variance in LLM baselines, we randomly sample 1 or 5 examples for each testing prompt. The examples come from the first batch of crowdsourced training data that are completely written and verified by humans. For GPT3.5, we use the default hyperparameters. Although it uses high top-p and temperature (both 1), the resulting generation diversity scores (dist-n) are still much lower than our methods.

H.3.1 Common Summary Prompt for GPT3.5

We test several different prompts and finally choose to use the following one because its zero-shot performance seems to be better than other prompts we tried.

> **Template H.1** *Please read the two stories below:*
> *"**{Story A Input}**"*
> *"**{Story B Input}**"*
> *Question: What is the common structure of the above two stories? Please be as specific as possible.*
> *Answer: The common structure of the stories is that "**{Summary Output}**"*

In Table 4, **GPT3.5 5 Shot (specific)** repeats the above template 5 times to conduct 5 Shot in-context learning. **GPT3.5 5 Shot (general)** replaces the word *specific* in the template with *general*. **GPT3.5 5 Shot (not mention)** removes the sentence *Please be as specific as possible.* in the template.

H.3.2 Story Generation Prompt for LLMs

We use the following prompt for GPT3, GPT3.5, OpenLLaMA, and Vicuna.

> **Template H.2** *Please generate a five-sentence daily-life story with around 50 words. The story should contain the following plot: **{Summary Input}***
> *"**{Story Output}**"*

For OpenLLaMA and Vicuna, they sometimes output new lines or still output other text after outputting ", so we did a post-processing to replace a new line with a space and keep only the text between two double quotation marks.

I Modeling Details

In this section, we describe some detailed design choices and the resources we used for training our models.

I.1 Reranker

Most of our crowd workers write valid common summaries, so we might not have enough negative examples from the summaries that fail to be implied by a story. To balance the ratio of positive and negative examples, we create pseudo-negative examples by switching the common summary in some randomly sampled positive examples with another similar common summary that is written for a different input story pair. When creating pseudo-negative examples for reranker, we use sentence BERT (sBERT), all-mpnet-base-v2[6] (Reimers and Gurevych, 2019), to measure the similarity.

I.2 From Short Story to Specific Summary

In our preliminary experiments, we found that the chosen LMs are significantly better than their smaller counterparts (i.e., T5-large and DeBERTa V3 base) and other alternatives we tried (i.e., Pegasus large (Zhang et al., 2020) and RoBERTa large (Liu et al., 2019)). For generating common summaries, we found that pretraining T5-3b using BookSum Paragraph dataset does not seem to be helpful, so we directly fine-tune the T5-3b using the crowdsourced data.

All the ROC stories we used in this paper come from the 2016 spring set. When we generate the summaries of every ROC story in Figure 2, we prepend "summarize: " prefix to the input to leverage the summarization ability of the T5.

I.3 Hyperparameters and Computing Resources

We use 8 NVIDIA V100 32GB GPUs to train our models. When the fine-tuning dataset is large, I would control the number of epochs to make the training finished within 2 days. My modeling codes are built on Huggingface and deepspeed[7].

[6] www.sbert.net and https://huggingface.co/sentence-transformers/all-mpnet-base-v2

[7] https://github.com/microsoft/DeepSpeed

When building the entailment hierarchy, we all set the k in top-k sampling as 50, which is the default value in Huggingface. Our reranker selects a story from 10 generated candidates.

We find that most of our models are not very sensitive to the learning rates and the number of training epochs given that the model does not overfit.

J Crowdsourcing Details for Dataset Construction

Writing common summaries is not only difficult for machines but also challenging for humans. MTurk workers are famously not reliable, so we design several tasks to achieve the following goals.

- All answers of the workers should be easily verifiable.
- The summaries and hypotheses should not be too specific or too general.
- We should minimize the crowdsourcing cost while maintaining the quality.

To achieve the goals, we provide example(s) in the instruction, write javascript to automatically check the responses of workers, let the workers check other workers' responses, continue filtering the workers and experimenting with different ways of collecting the entailment datasets. In total, we design 4 writing and verification tasks and collect 9 small batches to train our common summary generator. For generating the general summary/hypothesis, we launch 3 tasks and 6 small batches.

J.1 Quality Control

Our human annotation tasks are relatively complicated compared with most tasks on MTurk, so we spend time identifying the workers who are willing to read long instructions and answer the questions carefully. Since it is hard to find qualified workers and our tasks often require high cognitive loading, we control the hourly wage of the trusted MTurk workers to be around 14 US dollars after they are familiar with the task to ensure high data quality.

For the short story to specific summary mapping, we are able to utilize 95% of the responses. For the specific summary to general summary mapping, we are able to utilize 98.5% of the responses. We can keep the percentage high because we have a list of trusted workers from other projects and we stop some workers doing the tasks early once we find that the workers do not understand/follow the instructions.

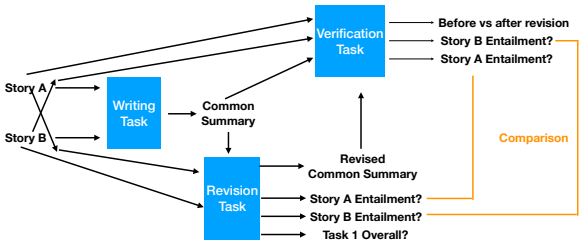


Figure 12: The workflow of writing and verifying the common summary for story A and story B.

J.2 From Short Story to Specific Summary

As illustrated in Figure 12, we design 3 crowdsourcing tasks to collect the common summary. The writing task in Figure 15 asks workers to write the common summaries. The revision task in Figure 16a asks different workers to verify and revise the written common summaries. If the summary is too general, make it more specific. If the common summary is not entailed by either input story, fix the summary. If the common summary is correct, provide another common summary that is also specific and very different from the existing one. The revision task could improve the quality, let us monitor the specificities, and help us collect more diverse and creative summaries. Finally, the verification task in Figure 16b asks different workers to verify and compare the summaries before and after the revision.

After we collect enough data to generate high-quality common summaries, we revise the verification task to become the fourth task in Figure 17 that asks the workers to verify the generated common summaries rather than the human-written summaries.

J.2.1 Monitoring the Specificities Common Summaries

In the revision task, the workers think 54% of the common summaries are good and they provide an alternative summary, 31% of the common summaries are not entailed by both stories and they provide a fix, and only 15% of the common summaries are too general and they provide a more specific summary.

In the verification task, among the revision for the "too general" common summaries, only 14% of the revision is labeled as more specific than the original "too general" common summaries (43% of the common summaries are labeled as paraphrases or almost the same as the "too general" common summaries, 14% are neutral, and 29% are actually

being more general).

The statistics show that

- only a small percentage of common summaries might have issues of being too general, and
- it is actually very difficult to make the common summary more specific via a revision process.

Therefore, we conclude that most of the workers try hard to follow the instruction to make the common summary as specific as possible, and getting too general summaries is not a major issue in our crowdsourcing process.

J.3 From Specific Summary to General Summary / Hypothesis

As we state in Appendix E, it is very hard to write a general summary/hypothesis for some similar specific summaries. To overcome the challenge, we generate five different common summary candidates for each input story pair, and the task in Figure 18 lets the MTurk workers select one summary from the candidates for writing the common hypothesis.

To ensure that the specific summary is more specific than the general hypothesis, we ask a MTurk worker to choose from implication, paraphrase, and others as the relation between the general hypothesis and specific summary. We only treat the implication as the positive entailment label in this work to make sure that the language models could generate more specific text at each iteration. Unlike writing the common summary for two input stories, it is much harder to write a very different alternative common hypothesis, so we skip the revision task and directly ask the workers to verify the written common hypotheses in Figure 19a and verify the generated hypotheses in Figure 19b.

J.4 Entailment Ambiguity

To Specific Summary: By comparing the responses from different workers, we find that some entailment labeling tasks have inherent ambiguity. For example, if the first short story in Table 1 says *Amy got a B+* instead of a straight A, some annotators would think getting a B+ is something impressive, but others might disagree. After the workers are familiar with the task, we found that around 20.8% of entailment judgments disagree with each other. When the two workers disagree, we label this ambiguous example as positive to increase the size of our training data for generating summaries.

To General Summary / Hypothesis: The relation between two sentences is sometimes ambiguous. For example, it is reasonable to annotate any of 3 options when the difference between the two sentences is small (e.g., "*A student got straight A in his/her school*" and "*Someone is a straight-A student*"). We observe that 37.0% of entailment judgments from a worker are different from the judgment from another worker. To reduce the number of training pairs with the paraphrase or other relations, we label this ambiguous example as negative (not entail).

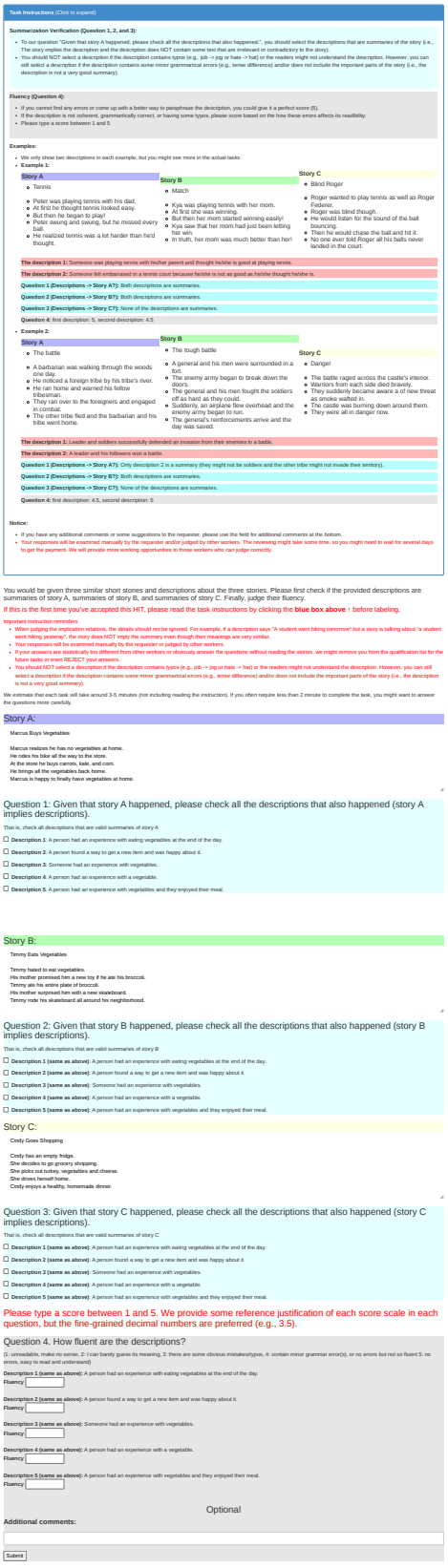


Figure 13: The evaluate task for common summary generation

(a) Short story

(b) Long story

Figure 14: The evaluation task for story evaluation. In each evaluation round, we will ask the worker to compare all stories generated by different methods. To reduce the task length, we only show the first two stories here.

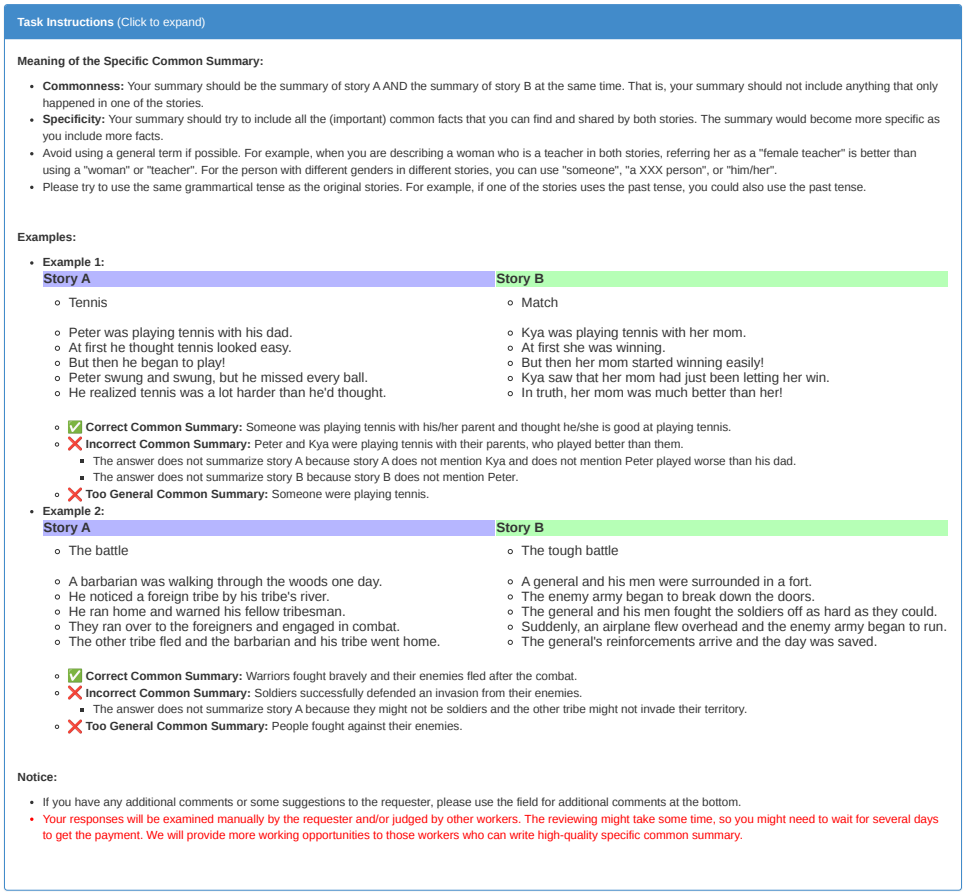
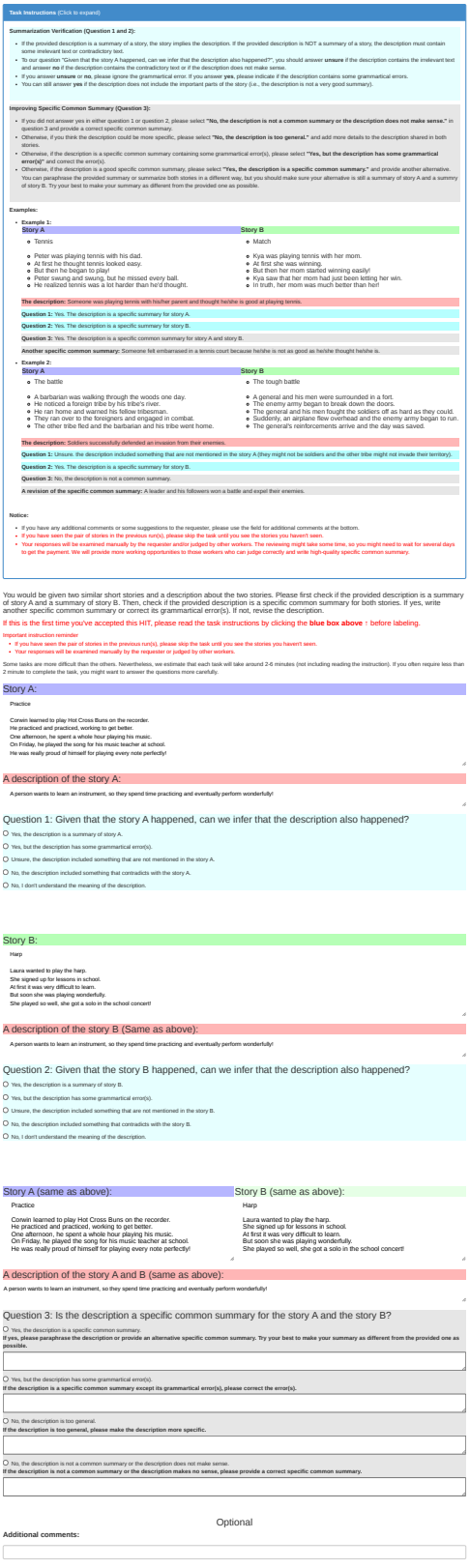
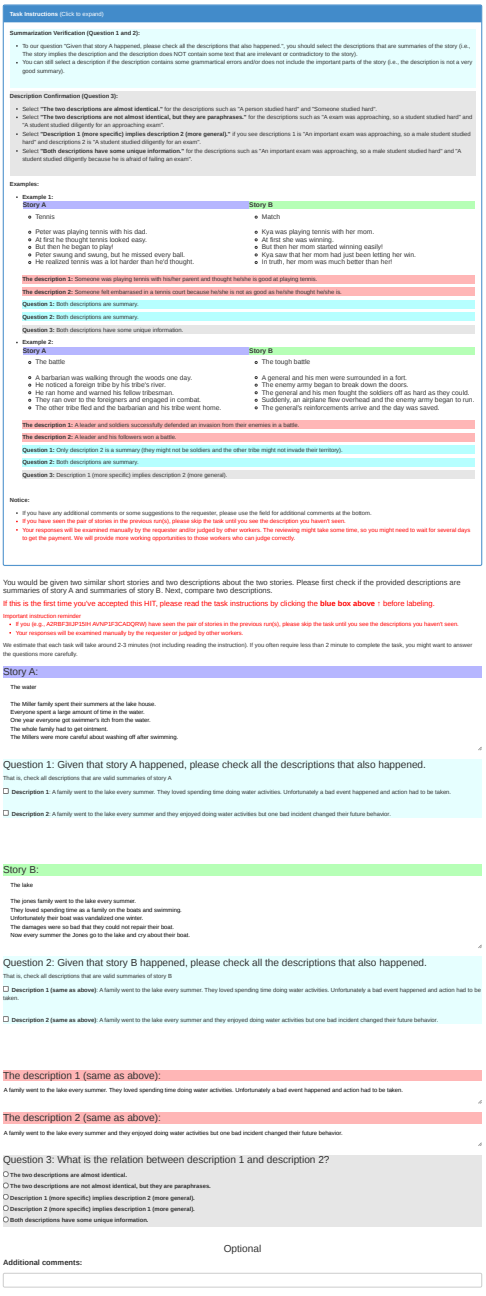


Figure 15: Our common summary writing task



(a) The revision task



(b) The verification task

Figure 16: Our common summary revision and verification tasks

Figure 17: Our task of verifying the generated common summaries

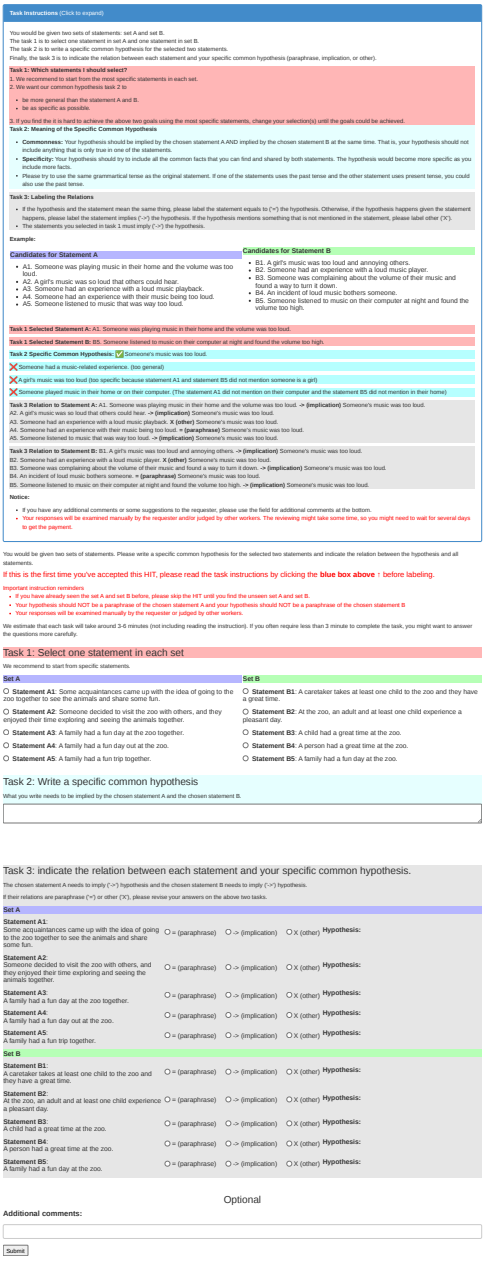


Figure 18: Our task of writing common hypotheses

(a) Verifying the written common hypotheses

(b) Verifying the generated hypotheses

Figure 19: Our task of verifying hypotheses