Neural Story Planning

Anbang Ye, Christopher Cui, Taiwei Shi, and Mark O. Riedl

Georgia Institute of Technology
{aye42, ccui46, maksimstw, riedl}@gatech.edu

Abstract

Automated plot generation is the challenge of generating a sequence of events that will be perceived by readers as the plot of a coherent story. Traditional symbolic planners plan a story from a goal state and guarantee logical causal plot coherence but rely on a library of hand-crafted actions with their preconditions and effects. This closed world setting limits the length and diversity of what symbolic planners can generate. On the other hand, pre-trained neural language models can generate stories with great diversity, while being generally incapable of ending a story in a specified manner and can have trouble maintaining coherence. In this paper, we present an approach to story plot generation that unifies causal planning with neural language models. We propose to use commonsense knowledge extracted from large language models to recursively expand a story plot in a backward chaining fashion. Specifically, our system infers the preconditions for events in the story and then events that will cause those conditions to become true. We performed automatic evaluation to measure narrative coherence as indicated by the ability to answer questions about whether different events in the story are causally related to other events. Results indicate that our proposed method produces more coherent plotlines than several strong baselines.[1]

Introduction

Automated plot generation is the challenge of generating a sequence of events that will be perceived by readers as the plot of a coherent story. Early solutions to story generation included symbolic planning (Meehan 1977; Lebowitz 1985; Porteous and Cavazza 2009; Riedl and Young 2010; Ware and Young 2011), and case-based reasoning (Pérez y Pérez and Sharples 2001; Gervás et al. 2005). These techniques explicitly represent and reason about the causal relationship between events. In particular, symbolic planners infer *causal relations* between events and through these causal relations, the logical soundness of a plan could be guaranteed. Story planners extended generic symbolic planners in a number of ways to create character believability (Riedl and Young 2010), character conflict (Ware and Young 2011), theory of mind (Ware and Siler 2021), etc. In story generators, logical soundness corresponds with the concept of *plot coherence* where each event—and the goal state—is justified by preceding events or the initial world state. Narrative psychologists note the importance of plot coherence in reader comprehension, avoiding confusion, acceptance of stories, and memory of stories (Trabasso and van den Broek 1985; Graesser, Lang, and Roberts 1991).

Unfortunately provable plot coherence comes at a cost. Symbolic planners require libraries of hand-crafted schemas that describe what actions (called events when planning stories) are available. Among other things, action schemas define *preconditions*—logical statements that must be true for an event to be executable—and *effects*—logical statements that describe how the state of the world is changed if an action succeeds in execution. While planned stories are highly causally coherent, the reliance on hand-crafted libraries of action schemas and pre-existing symbols for characters, objects, and locations, limits the length and diversity of plots.

Neural language model based approaches to story generation, on the other hand, can generate more diverse stories about any number of topics included in the training data (Martin et al. 2017; Roemmele and Gordon 2018; Fan, Lewis, and Dauphin 2018; Yao et al. 2019; Goldfarb-Tarrant, Feng, and Peng 2019; Rashkin et al. 2020). This is especially true for large, pre-trained neural language models that have been trained on, amongst other things, large corpora of stories. Neural story generators create stories by sampling from the probability distribution $P(w_n|w_1, w_2, ..., w_{n-1}; \theta)$ where θ are the parameters of the language model. Sampling from this learned distribution emulates human patterns of language, including that found in stories. However, when addressing novel combinations of topics and circumstances sampling is not guaranteed to provide causal coherence. Other limitations of neural text generation, such as repetition and generic responses, have also been documented (Holtzman et al. 2019).

In this paper, we seek to unify the strengths of symbolic planning and neural text generation to achieve causally coherent, ending-guided, story plotlines. To this end, we have developed a story planner based on *partial-order causal link planning* (POCL) (Penberthy and Weld 1992) to generate story plotlines. However, instead of using a library of hand-crafted action schemas and world state symbols, our planner uses a large pre-trained language model (specifically GPT-J-6B) to infer events and their preconditions. In this way, the story planner is able to operate without pre-specifiying

[1] Our source code will be made available at: https://github.com/YeAnbang/Neural-Planner

actions, characters, world locations, or objects in the world. Yet the planner can still identify causal relations between events and ensure causal coherence of stories.

Our planner works via ends-means chaining—working backward from a given story ending. Given a text description of the ending of the story as well as an optional set of initial-state conditions that describe elements of the story world that can be assumed true, the system queries the language model to generate preconditions that must be satisfied for the given ending to occur. For example (see Figure 1), the event, "Sally buys a gun from the gun store" will produce preconditions that include "Sally is at the story" and "Sally has money". Each of these preconditions describe a partial world state that must come into existence through the effects of preceding events. Next, the system queries the language model for an event that would bring about each precondition. Continuing the example, "Sally is at the store" produces a new event: "Sally walked to get to the store". The process repeats, resulting in a graph—a partially-ordered plan of events—that can be topologically flattened into the plot line of a story.

Our system outperforms other strong baselines in generating coherent plotlines as determined by ability to answer questions about the causal relations between events. The ability to answer questions about stories is an indicator of support for reader comprehension (Graesser, Lang, and Roberts 1991).

Background and Related Work

Formally, a planner finds a sequence of actions that transforms the initial world state into one in which a goal situations holds. Story and plot generation using symbolic planning (Meehan 1977; Lebowitz 1985; Porteous and Cavazza 2009; Riedl and Young 2010; Ware and Young 2011; Ware and Siler 2021) is known to result in causally coherent plotlines but at the expense of story diversity and length due to reliance on hand-crafted action schemas and symbols.

Partial Order Causal Link Planning

Several story planners (Riedl and Young 2010; Ware and Young 2011) use a form of symbolic planning called *partial order causal link planning* (POCL) (Penberthy and Weld 1992). POCL planners search plan-space where every node in the space constitutes an unique partially-ordered plan and the planner transitions from one plan to the next by adding an action or resolving a logical causal flaw due to partial ordering. A plan is represented as a tuple $P = \langle A, C, O \rangle$. A is a set of actions instantiated from a library of action schema templates that specify preconditions effects. C is a set of *causal links* of the form $a_1 \xrightarrow{c} a_2$ where $a_1, a_2 \in A$ and c is a predicate condition that unifies with an effect of a_1 and a precondition of a_2. O is a set of temporal ordering constraints $a_1 \rightarrow a_2$ that indicates that a_1 must be temporally ordered before a_2.

At every iteration, the planner selects a plan on the fringe of the plan-space and an action in that plan with an unsatisfied precondition (or the goal state). An unsatisfied precondition is one in which there does not exist a causal link that points to the action that has a matching the condition. A new successor plan is generated for each way of satisfying the action precondition—either by adding a new action and causal link, or by identifying an existing, preceding action with a matching effect (or the initial state)—and extending a causal link between them. Other operations not summarized here ensure logical soundness. Planning terminates when a plan is found with no actions with unsatisfied preconditions.

While there are newer, more computationally efficient symbolic planners, we start with POCL planners for three reasons. First, they explicitly identify preconditions that must be satisfied and explicitly identify actions that satisfy preconditions, making plans explainable. Second, the explicit representation of actions and preconditions makes it amenable to adaptation to the use of LM inferences. Extending our proposed approach to more modern planning algorithms is left for future extensions of this work. Third, POCL planners are cognitively plausible ends-means planning processes for humans (Young 1999).

Neural Story Generators

Early attempts at neural story generation used language models trained on story corpora to generate stories (Roemmele and Gordon 2018) or plots (Martin et al. 2017). However, such vanilla use of language models were found to have trouble maintaining coherent context. Guan et al. (2020) further fine-tunes a pre-trained language model on commonsense datasets to help increase the coherence of stories. Attempts to control the coherence of stories and plots include conditioning (e.g., Rashkin et al. (2020)) or hierarchical generation (Fan, Lewis, and Dauphin 2018; Yao et al. 2019; Fan, Lewis, and Dauphin 2019). Hierarchical generation is similar to greedy planning by first generating a "sketch" or "skeleton" at a higher level of abstraction.

Language models are not generally aware of, or able to drive toward, a given story ending. Tambwekar et al. (2018) fine-tuned language models to be goal-aware. The EDGAR system (Castricato et al. 2021) generates stories backward using language model based question-answering to iteratively ask how the story state came to be. This is similar to our approach, but does not explicitly reason about causality.

C2PO (Ammanabrolu et al. 2021) conducts a bi-directional search, querying story events that can possibly follow or possibly precede events using the COMET (Bosselut et al. 2019) model of commonsense inference. C2PO is the closest to our approach in terms of conducting a search through event space. C2PO relations between events are referred to as *soft* causal links because the capture inferred (probabilistic) event relationships, whereas our system uses *hard* causal links that capture explicit conditional relations between events as in POCL planning.

The TattleTale system (Simon and Muise 2022) uses a classical symbolic planner to generate a totally ordered sequence of actions and then uses this sequence to condition a language model to generate natural language stories. The symbolic planner uses hand-crafted action schemas.

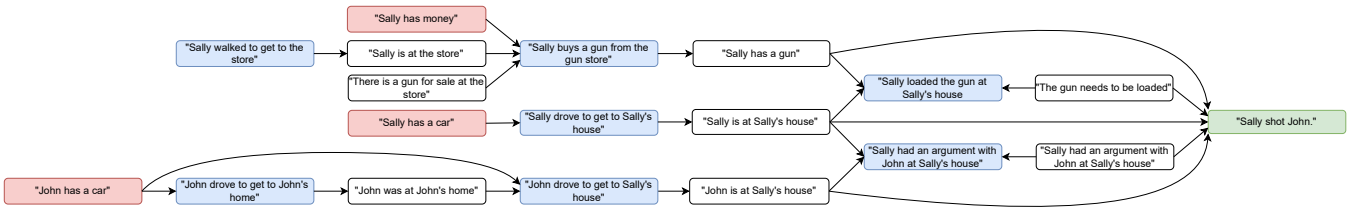

Figure 1: Visualization of a partial order plan for a plotline. The final event, which is given as input, is green. Events are blue. Preconditions are white and red; red preconditions are those that matched against given initial conditions. Arcs from events to preconditions are *causal links*, indicating that the event makes the condition true. Arcs from precondition to event indicate that the former is a necessary condition of the event. The plan is generated from right to left, starting with the final event. The plan can be totally-ordered, as shown in Table 1.

Ending: Sally shot John
Sally walked to get to store.
Sally buy a gun from the store.
Sally drove to get to Sally's house.
Sally loaded the gun.
John drove to get to John's home.
John drove to get to Sally's house.
Sally had an argument with John at Sally's house.
Sally shoot John.

Table 1: An example of ending-guided planning using neural planner. Events from the plan in Figure 1 are shown in chronological order.

Neural Story Planner

Our proposed technique generates causally coherent story plans using the concepts of causal links from POCL planning. However, unlike POCL planning, events and their preconditions are inferred by a large language model (LM) and represented as text descriptions. An example plan generated by our system is shown in Figure 1 with corresponding total-ordered plot in Table 1.

Action preconditions can be thought of as commonsense rules for action applicability. Commonsense knowledge refers to commonly held beliefs about our physical world. Because our planner works in an open world setting we must infer what preconditions need to be satisfied to enable an action, and what actions can be taken to satisfy those preconditions. Language models above 1B parameters have been demonstrated to capably answer commonsense questions (Bisk et al. 2020) and we use the GPT-J-6B[2] language model to infer both events and preconditions.

Our planner starts with an input sentence s that specifies the ending of a story. The planner is also provided with a set of initial conditions I, which are sentences describing facts known to be true about the world before the story begins. The system then recursively uses the LM to infer preconditions of any event in the plot, beginning with s and, for each precondition, uses the LM again to infer an event that would cause that condition to become true. See Algorithm 1.

The algorithm is a variation on POCL planning, with a few modifications. For any given precondition, we semi-greedily accept the first event inferred by the LM. The open-world setting is such that the logical impossibility that happens in closed worlds with fixed symbols rarely happens and we find the exhaustic plan-space expansion to be unnecessary. However, some inferences result in cycles of repeated events, in which case the planner backtracks and selects alternative inferences.

[2]https://github.com/kingoflolz/mesh-transformer-jax

Precondition Generation

We use the few-shot capabilities of large language models to infer the preconditions of events. We designed prompts based on psychological theories of reading comprehension that indicate that readers actively track different types of relations between events (Trabasso and van den Broek 1985; Graesser, Lang, and Roberts 1991). *Consequence* relations are equivalent to causal links and indicate that the latter event cannot occur unless the former occurs. Thus consequence relations also capture whether one event enables another to be possible. *Reason* relations capture the goals of characters that explain why they are doing something. *Outcome* relations capture which events satisfy which goals. *Initiate* relations capture when an event causes a character to form a goal.

We break preconditions into six different classes and have developed a prompt for each. The first four classes capture *consequence* relations:

- **Item Need.** What item(s) must the character possess for an action to execute? E.g., in order for one character to drive somewhere they must have a car.

  What item must $[Person]$ possess to $[Action]$?
  Answer: $[Person]$ must have $[Answer]$

- **Location.** Where must the character be in order to carry out an action? E.g., for one character to buy something from a store the character must be in the store.

  $[Context]$
  $[Event]$.
  $[Optional\ Hint]$
  Where is $[Person]$? : $[Answer]$

  The context consists of all events that are ancestors of the current event; even though they will come later in the story, they bias the LM toward reusing locations. The

hint, which is optional, is filled with any location information that is heuristically extracted from the event.

- **Item state.** What must be true about an item for an action to be performed? E.g., in order to shoot someone, the gun must be loaded.

  $[Event]$
  What can we tell about the $[item]$ other than how $[Person]$ obtained it? $[Answer]$

- **How.** What needed to have happened for something to be achievable now? E.g., "John became a pro wrestler" would have a how-precondition that could be satisfied by another event such as "John trained for one year".

  $[Context]$
  $[Event]$
  How does $[Event]$? $[Answer]$

The next precondition class aligns with *initiates* relations. Our version focuses on character-character interactions.

- **Interactions with others.** What must have happened between two characters for an action to be peformed? For example, a character might perform a violent action toward another character if first there was an argument.

  $[Context]$
  $[Event]$
  Prior to $[Event]$, $[Answer]$

The final class of precondition is the *reason*, which can be satisfied by an event that causes a character to have a goal:

- **Reason.** What serves as the reason for an event? E.g., the event "John was arrested" would have a reason-precondition that could be satisfied by an event such as "John stole a car".

  $[Context]$ $[Event]$ because $[Answer]$

Each prompt consists of at least five few-shot examples and the last leaves the answer blank to be completed by the LM.

When the inference is a short phrase, as is the case in all of our precondition classes, but especially true for *Item Need* preconditions, querying a LM suffers from surface form competition (Holtzman et al. 2021) in which very common responses can be favored over those that are more responsive to the prompt. For example, to the question "What does John need to clean the table?" a LM would prefer the answer "nothing" over the correct answer "cleaning cloth" because GPT-J has seen the former more often than the latter in its training corpus. We adapt the *Domain Conditional Pointwise Mutual Information* (PMI_{DC}) rescoring method of Holtzman et al. (2021) to address the surface form competition issue. We use $PMI_{\text{DC}} = \frac{P(y|x,domain)}{P(y|domain)}$, where y is the candidate and x is the input sentence. We observe that PMI_{DC} rescoring only work robustly when all options are reasonable to some extent. We filter out low frequency candidate before calculating PMI_{DC} scores. An exception to the above is the *location* precondition class, where the answer is biased by context and hints. In this case, we do rescore with PMI_{DC} and simply take the most frequent candidate.

As the LM can sometimes produce outputs that are semantically identical but are only slight lexical variations of each other, we also apply a basic cosine similarity check to filter out these duplicates. We use different thresholds depending on the precondition type. For *Item Need* and *Location* we use a threshold of 0.75 due to their shorter generations. For everything else, the default threshold is 0.8.

Not every event requires all six classes of precondition; forcing the LM to generate preconditions when they are not needed and will not make sense can result in unpredictable responses. We apply heuristics to determine when a precondition type is unnecessary. First, an *interaction with others* precondition is only necessary when an event has two peoples' names. Second, *reason* and *how* preconditions are necessary only when an event can be expanded syntactically by adding "because $[X]$" or "through $[X]$", respectively, where X is some sentence continuation. For example, "John was arrested *because* John stole a car" or "John became a pro wrestler *through* training hard". We include few-shot examples that contain both positive examples (completion examples that contain keywords), negative examples (completion examples that do not contain keywords) and detect the appearance of keywords in text completion results.

We do not require *reason* preconditions that contain the terms "want" or "need". Reason preconditions that contain those two keywords usually show the intention of the character, e.g., "John wants to buy a gun" as a precondition for the action "John walked to the store". Unlike *reason* preconditions that help to develop the plot by introducing a new event, e.g., "John's old glass broke" in response to the action "John bought the new glasses", intentions are generated by looking into the existing part of the story and therefore do not help with expanding the story plot. Want and need reason-preconditions are deleted automatically.

We require every action to have a *location* precondition. Our few-shot examples give examples of responses of "nothing", which we treat as a keyword to indicate that an event shouldn't be considered as having a location. When we detect the nothing response, we delete the precondition. we use the same strategy with the *item state* precondition.

Event Generation

In POCL-style planning, preconditions are satisfied by the effects of events. We directly infer events such that they satisfy a precondition. For *how*, *interaction with others*, and *reason* preconditions, we directly copy the sentence from the precondition to make a new event because the nature of those precondition inferences produce sentences that are also events. These suggest pairs and chains of events that go together naturally, similar to *scripts* (Schank and Abelson 2008). For *item need*, *location* and *item state* preconditions we use specialized prompts for each precondition type:

- **To satisfy *item need* precondition:** What event occurred so that the character obtained the required item? E.g., the *item need* precondition "John has gun" could be satisfied by "John bought a gun from the store".

  Context: $[Sentence]$
  How did $[Person]$ get $[item]$
  Answer: $[Answer]$

- **To satisfy *item state* precondition:** What event need to happen to change the state of an item to its desired state? E.g., the *item state* precondition "gun is loaded" would be true after "John loaded the gun".

  Sentence: $[Sentence]$
  $[item\ state]$ $[Answer]$

- **To satisfy *location* precondition:** What event need to happen for the character to arrive at the desired location? E.g., "John drove to get to John's house" causes "John at John's home" to be true.

  Context: $[Sentence]$.
  How did $[Person]$ get to $[location]$?
  Answer: $[Answer]$

As with precondition generation, each prompt has seven examples and the eighth has the answer blank. When an event is generated to satisfy a precondition on another event, we create a *causal link*, which is a directed arc from the new event to the precondition, indicating that (a) the precondition is satisfied, and (b) the new event is a temporal predecessor. The plot is thus represented as a directed acyclic graph, as shown in Figure 1.

We make a greedy assumption that if more than one event has the same precondition then the same event will satisfy both preconditions. We define two preconditions to be the same if (1) they are associated with the same character, and (2) the cosine similarity of the two preconditions surpass certain threshold (0.75 for *item need* and *location* and 0.8 for other categories). Thus, before generating a new event, we check to see if an existing event can be used and a causal link extended from the already existing event to the precondition. This corresponds to the concept of action re-use in POCL planning. The exception to this rule is when the new causal link causes a cycle in the graph.

When duplicate events are generated, as determined by cosine similarity, we record an event has having multiple possible phrasings. The phrasing with the highest count is presented as part of the plan. Ties are broken by taking the phrase with the lowest perplexity. Additionally, we also discard the generated precondition if it is identical to its parent event. Note, we do not do this for the reverse, where the generated event is identical to the precondition, as shown in Figure 1 in the "Sally had an argument with John in Sally's house" precondition and event.

Preconditions that match the initial state conditions I are never used to generate events since these are conditions given by the user.

We do not model negative effects of events. In POCL planning, it is possible for an action to have an effect that negates a precondition. In that regard, our system is similar to the graph expansion stage of graph plan (Blum and Furst 1997) without the forward-checking phase. We do this for two reasons. First, our planner operates in an open-world and, because preconditions are inferred by a LM, we cannot guarantee we have the complete set of preconditions needed to ensure true logical soundness like in a POCL planner. Second, stories are relatively permissive of missing events when readers can themselves infer missing details and events. For example, if John has an item and then Sam has the item later, the reader doesn't need to be told that John no longer has the item, even though our planner doesn't explicitly represent it.

Algorithm 1: Neural Plot Planner

Input: ending event sentence g; Initial conditions I.
Initialize a plan $P \leftarrow \emptyset$; Initialize $queue \leftarrow \{g\}$.
while $queue \neq \emptyset$ **do**
  Let $event \leftarrow$ **pop**$(queue)$
  Let $context \leftarrow$ sequence of events collected by running a breadth-first search from $event$ to g.
  Let $\Lambda \leftarrow$ all satisfied preconditions
  Let $\Gamma \leftarrow generate\ preconds(event)$ for each character in $event$
  if adding any precondition in Γ creates a cycle **then**
    remove $event$ from P.
    $\Gamma \leftarrow$unsatisfied preconditions due to removing $event$.
  for each $c \in \Gamma$ **do**
    if $c \in I$ or c meets conditions for not being expanded **then**
      $P \leftarrow P \cup \{nil \xrightarrow{c} event\}$ ▷ Dangling precondition
    else if there exists a precondition $c' \in \Lambda$ that is similar to c **then**
      $event' \leftarrow$ event that satisfies c'
      $P \leftarrow P \cup \{event' \xrightarrow{c'} event\}$ ▷ Reuse precondition
    else
      $event' \leftarrow generate\ event(c, context)$
      $P \leftarrow P \cup \{event' \xrightarrow{c} event\}$ ▷ Satisfy with new event
      $queue \leftarrow queue \cup \{event'\}$

The open world setting combined with surface form competition introduces the possibility of cycles where event e_1 results in precondition p_1 is semantically equivalent to another precondition p_2 that is an ancestor (p_2 is on a causal path from the goal to to p_1). When this happens, we remove e_1 from the plan and backtrack to re-generate that event.

The planner terminates when there are no unsatisfied preconditions. The algorithm is shown in Algorithm 1.

Final Total Ordering

When there are no further preconditions to satisfy, the final step is to produce a total ordering of the events in the plot graph. In POCL planning, any two events that are not ancestors or descendant of each other are considered partially ordered and any topological sort that doesn't violate any temporal ordering constraints is valid. Because of our open-world assumptions, we implement a topological sort that prefers certain orderings over others. Whenever there are two events e_i, e_j that are neither descendants nor ancestors, we order them $e_i > e_j$ when $type(e_i) > type(e_j)$ and $type(\cdot)$ is a function that maps an event to the type of precondition it satisfies and *reason* $>$ *how* $>$ *item_needed* $>$ *item_state* $>$ *location* $>$ *interaction_with_other_person* $>$ *current action*. The greater-than symbol means earlier in the totally ordered sequence. Following cognitive science (Graesser, Lang, and Roberts 1991), events that initiate character intentions should precede other actions that might relate to the pursuit of those intentions. How things came to be, should also precede events that make use of those establishing events. Character interactions, especially conflict, are often the culmination of many establishing events.

Evaluation

We evaluate the extent to which our planning technique generates coherent plots. Psychological studies of human reader comprehension (Graesser, Lang, and Roberts 1991) measure reader comprehension by reader ability to answer questions about a story. In our setting, we prompt GPT-3 (Brown et al. 2020) to answer *enablement* questions about generated plots.[3]

We use GPT-3 (text-davinci-002) as an estimate of the ease by which questions can be answered about generated stories, and thereby an estimate of the extent to which generated stories support question-answering for reader comprehension.

We evaluated our system and all baselines by generating plotlines that were seeded from randomly chosen stories from the test split of ROCStories dataset (Mostafazadeh et al. 2016). The ROCStories dataset contains plot-like common everyday stories with titles for each story. We randomly sampled 50 story titles from the test split of ROCStories to seed our system and the baselines. We generate a story plotline for each title and then assess whether the plotline is coherent. Systems are scored by the percentage of the time that GPT-3, appropriately prompted, can determine that a given event in a story is enabled by any prior events in the story.

We do not evaluate systems using perplexity or variants of BLEU score. Perplexity and BLEU compare system outputs to expected outputs in a supervised dataset. Sentences in a story can be generated that are considered good even if they do not match—or have overlapping tokens—with a ground-truth story. Further, our method does not sample from the language model in the conventional way.

Coherence Measure

To measure whether the generated stories support enablement comprehension, we pose questions to GPT-3 in the form "What's the action, if any, enabled *ACTION*?" along with the sentences from the earlier parts of the story plan for GPT-3 to choose from. Since the first sentence of a story is not enabled by default, we excluded the first sentence in our evaluation. We set $temperature = 0.7$.

We consider an enablement question *answerable* if the answer is found in the earlier part of the story and is different from (percentage of overlapped 1-gram ≤ 0.7) the queried event. If the question yields no answer, GPT-3 will respond with "none". We include negative few-shot examples in the prompt that are answered with "none" so that GPT-3 knows it is a valid response.

$$score = \frac{\#answerable\ enablement\ questions}{\#questions} \quad (1)$$

To validate our measure, we apply the measure to 100 stories randomly sampled from the test split of the ROCStories dataset. Stories in the dataset are assumed to be coherent. In half of the stories, we force incoherence by randomly replacing the queried event with a random even taken from a different story in the dataset. That is, half of the stories should be answerable and half of the stories should produce "none" when prompted with our above enablement prompt. Our measure yields an 85.39% response rate on unaltered stories and correctly predicts "none" 71% of the time when presented with an incoherent story. Thus our measure has an overall accuracy of 78.20%. ROCStories do not always contain explicitly enabling events. It is also sometimes possible for a randomly inserted event to appear to have an enabling predecessor action. Thus some degree of error is expected, and we deem our measure to be sufficient to produce a relative assessment between plot generation systems that produce story plotlines with similar styles.

[3]GPT-3 has been used to evaluate other systems such as summarization (Goyal, Li, and Durrett 2022).

System	Before filtering	After filtering
Ours	4.16	4.08
C2PO	5.62	5.42
plan write revise	5.00	4.22
comGen	4.98	4.35
GPT-J	3.86	3.63
ROC	5.00	4.56

Table 2: Average number of sentences for each system before and after pre-processing to remove non-event sentences. This shows that systems are comparable in length when prepared for evaluation.

Models and Baselines

We evaluated the performance of our neural planner against four baselines:

- *GPT-J-6B*, which is the same model we use to infer preconditions and events, but allowed to generate an entire story. To induce it to generate plots at a comparable level of abstraction to our system, we prompt it with few-shot examples pulled randomly from the ROCStories dataset. The prompt used in this baseline can be found in Appendix A.2. We set the temperature to 1.2.
- *C2PO* (Ammanabrolu et al. 2021), which uses bidirectional interpolation using commonsense inference of events. We use the first and last sentence of GPT-J-6B generated stories (see above) in our evaluation set.
- *comGen* (Guan et al. 2020), a transformer based language model fine-tuned on ROCStories and two commonsense knowledge bases. We sampled from the provided stories used in their evaluation to use as a baseline.
- *plan-write-revise* (Goldfarb-Tarrant, Feng, and Peng 2019) fine-tuned on ROCStories with titles as the topic.

To ensure we evaluate on only *plot*, which are events that change the world state (Prince 2003), we parse the outputs of the baselines to remove non-action sentences that give declarative statements about fact about the story world and characters, e.g., "Sam was a high school wrestler." We also filtered outputs that consisted of only one sentence, as we cannot fairly compare the logical coherence of a single-event plot against plot lines consisting of multiple events. We provide average generation lengths for each baseline, before and after filtering in Table 2.

System	enablement(%)
Ours	**85.71**
C2PO	75.11
plan write revise	67.70
comGen	76.65
GPT-J	71.43
ROCStories	85.39

Table 3: The Neural Planner achieves the highest percentage in terms of answerable enablement questions compares to other baselines.

Note that none of the baselines we compare to are ending-guided like our system. C2PO, the closest, uses bidirectional interpolation. Other baselines are conditioned on leading context. However, our evaluation indicates that the generated plot line's length does not appear to have any significant effects on our evaluation metrics.

For our system, we require an ending event sentence. We first prompt GPT-J-6B to generate an ending based on the provided title. Then, we use the ending to generate a story plot. Our model is capable of generating more than one object precondition for a given event. To make our system more comparable to baselines, which are more linear in generation process, we constrain the item-precondition generation to a single precondition.

This is roughly in line with what we observed with the ROCStory dataset, as the majority of events in that dataset tend to involve only one interact-able object.

We didn't evaluate against other story generators such as Goldfarb-Tarrant et al. (2020) or Castricato et al. (2021) because these models tend to generate stories with a lot of dialogue and many descriptive scenes. This creates difficulty when evaluating the logical coherency of those systems' outputs as well as extracting the distinct plot lines to use as a baseline. For consistency of evaluation, we only include methods with outputs that have a similar style and structure as our story plots.

Results and discussion

The results are shown in in Table 3. Our planner achieves the highest percentage of answerable enablement questions by a considerable margin. This can be partially explained by the fact that our planner is intentionally introducing events to the story that enable future events. The preconditions and linkages between events are not present in the final rendering of a plotlines and any enablement relations are still implicit. Our evaluation metric is likely sensitive to this type of structure. In that sense, the metric can be seen as a measure of the ease with which enablement questions can be answered about a story.

C2PO and comGen use commonense inferences from COMET (Bosselut et al. 2019) that have the potential to create explicit enablement relations. C2PO in particular generates commonense inteferences about "wants", what is expected to come after an event, and "needs", what is expected to precede each event in a story. As noted by Ammanabrolu et al. (2021) these commonsense relations between events are very similar to causal links though the exact nature of the relation is implicit.

We repeat the ROCStories response rate of 85.39% in Table 3 for completeness. The stories in the ROCStories dataset are coherent but not necessarily written to make answering enablement questions easy to answer—events may be omitted because they are considered obvious. Therefore, this comparison is not on a level playing field, and our results do not necessarily imply that our planner is performing above human level.

Conclusions

In this paper, we present a novel use of neural language models for generating story plotlines by unifying language models with partial order, causal link planning. Our system chains backward from a given ending of a story, and reasons about the conditions that are necessary for each event to occur. It generates events that causally enable subsequent events while backward chaining. In order to operate in an open world domain and be able to generate stories without predetermined actions schemas or characters, our story planner uses the large, pre-trained language model to infer the conditions and events.

This work suggests that pre-trained language models provide affordances for generating coherent narrative content other than generating continuations from a single prompt. Specifically, a large pre-trained language model can operate as a commonsense knowledge base about event preconditions and ways to bring about world conditions when prompted to do so. While our technique makes heavy use of hand-crafted prompts, all generation is coming from the same model guided by a search algorithm inspired by classical planning. The ability to construct a search space provides a number of benefits including causal relations between events that correlation with improved coherence and reader comprehension. It also provides a principled means of reasoning about what sentences should occur in a plot-like story beyond reliance on statistical sampling.

POCL plans are interpretable. One can look at any action in a plan and, through causal links, know why the action is present in the plan. Causal links indicate which actions are necessary for other actions to execute, and, more importantly, how each action contributes to the achievement of the goal. Our technique also inherits a degree of interpretability—despite the use of neural language models—because our plan data structures also contain causal links. We cannot guarantee that conditions, and thus links, are missing. However, the plan structure generated by our technique provides insights into the decisions that the planner was making during generation.

Our results indicate that story plotlines generated by our planner are coherent as measured by the ability to answer questions about causal enablement relations in generated stories. Like POCL planning, generated stories focus on physically grounded action. One can consider generated plotlines as the "skeletons" (Simon and Muise 2022) for fully fleshed out natural language stories that can include details and dialogue generated later. This work presents a step forward toward the open research challenge of generating

stories in open-world that are also guaranteed to be coherent and comprehensible.

References

Ammanabrolu, P.; Cheung, W.; Broniec, W.; and Riedl, M. O. 2021. Automated Storytelling via Causal, Commonsense Plot Ordering. In *AAAI*.

Bisk, Y.; Zellers, R.; Bras, R. L.; Gao, J.; and Choi, Y. 2020. PIQA: Reasoning about Physical Commonsense in Natural Language. In *AAAI Conference on Artificial Intelligence*.

Blum, A. L.; and Furst, M. L. 1997. Fast planning through planning graph analysis. *Artificial Intelligence*, 90(1).

Bosselut, A.; Rashkin, H.; Sap, M.; Malaviya, C.; Celikyilmaz, A.; and Choi, Y. 2019. COMET: Commonsense Transformers for Automatic Knowledge Graph Construction. In *ACL*.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*.

Castricato, L.; Frazier, S.; Balloch, J.; and Riedl, M. 2021. Tell Me A Story Like I'm Five: Story Generation via Question Answering. *Proceedings of the 3rd Workshop on Narrative Understanding*.

Fan, A.; Lewis, M.; and Dauphin, Y. 2018. Hierarchical Neural Story Generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Fan, A.; Lewis, M.; and Dauphin, Y. N. 2019. Strategies for Structuring Story Generation. *CoRR*, abs/1902.01109.

Gervás, P.; Díaz-Agudo, B.; Peinado, F.; and Hervás, R. 2005. Story Plot Generation Based on CBR. *Know.-Based Syst.*, 18(4–5).

Goldfarb-Tarrant, S.; Chakrabarty, T.; Weischedel, R. M.; and Peng, N. 2020. Content Planning for Neural Story Generation with Aristotelian Rescoring. *CoRR*, abs/2009.09870.

Goldfarb-Tarrant, S.; Feng, H.; and Peng, N. 2019. Plan, Write, and Revise: an Interactive System for Open-Domain Story Generation. *CoRR*, abs/1904.02357.

Goyal, T.; Li, J. J.; and Durrett, G. 2022. News Summarization and Evaluation in the Era of GPT-3.

Graesser, A. C.; Lang, K. L.; and Roberts, R. M. 1991. Question answering in the context of stories. *Journal of Experimental Psychology: General*, 120.

Guan, J.; Huang, F.; Zhao, Z.; Zhu, X.; and Huang, M. 2020. A Knowledge-Enhanced Pretraining Model for Commonsense Story Generation. *CoRR*, abs/2001.05139.

Holtzman, A.; Buys, J.; Forbes, M.; and Choi, Y. 2019. The Curious Case of Neural Text Degeneration. *CoRR*, abs/1904.09751.

Holtzman, A.; West, P.; Shwartz, V.; Choi, Y.; and Zettlemoyer, L. 2021. Surface Form Competition: Why the Highest Probability Answer Isn't Always Right. In *Proceedings of EMNLP 2021*.

Lebowitz, M. 1985. Story-telling as planning and learning. *Poetics*, 14.

Martin, L. J.; Ammanabrolu, P.; Hancock, W.; Singh, S.; Harrison, B.; and Riedl, M. O. 2017. Event Representations for Automated Story Generation with Deep Neural Nets. *CoRR*, abs/1706.01331.

Meehan, J. R. 1977. TALE-SPIN, An Interactive Program that Writes Stories. In *IJCAI*.

Mostafazadeh, N.; Chambers, N.; He, X.; Parikh, D.; Batra, D.; Vanderwende, L.; Kohli, P.; and Allen, J. F. 2016. A Corpus and Evaluation Framework for Deeper Understanding of Commonsense Stories. *CoRR*, abs/1604.01696.

Penberthy, J. S.; and Weld, D. S. 1992. UCPOP: A Sound, Complete, Partial Order Planner for ADL. In *KR*.

Porteous, J.; and Cavazza, M. 2009. Controlling Narrative Generation with Planning Trajectories: The Role of Constraints. In *ICIDS*.

Prince, G. 2003. *A Dictionary of Narratology*. Lincoln, Nebraska: University of Nebraska Press. Hey Babe, Take a Walk on the Wild Side–Creative Writing in Universities.

Pérez y Pérez, R.; and Sharples, M. 2001. MEXICA: A computer model of a cognitive account of creative writing. *J. Exp. Theor. Artif. Intell.*, 13.

Rashkin, H.; Celikyilmaz, A.; Choi, Y.; and Gao, J. 2020. PlotMachines: Outline-Conditioned Generation with Dynamic Plot State Tracking. *CoRR*, abs/2004.14967.

Riedl, M. O.; and Young, R. M. 2010. Narrative Planning: Balancing Plot and Character. *Journal of Artificial Intelligence Research*, 39.

Roemmele, M.; and Gordon, A. 2018. An Encoder-decoder Approach to Predicting Causal Relations in Stories. In *Proceedings of the First Workshop on Storytelling*.

Schank, R. C.; and Abelson, R. P. 2008. *Scripts, plans, goals and understanding: An inquiry into human knowledge structures*. Psychology Press, Taylor et Francis Group.

Simon, N.; and Muise, C. 2022. TattleTale: Storytelling with Planning and Large Language Models. In *ICAPS Workshop on Scheduling and Planning Applications*.

Tambwekar, P.; Dhuliawala, M.; Mehta, A.; Martin, L. J.; Harrison, B.; and Riedl, M. O. 2018. Controllable Neural Story Plot Generation via Reinforcement Learning. *arXiv: Computation and Language*.

Trabasso, T.; and van den Broek, P. 1985. Causal thinking and the representation of narrative events. *Journal of Memory and Language*, 24(5).

Ware, S.; and Young, R. M. 2011. CPOCL: A narrative planner supporting conflict. In *Proceedings of the 7th AAAI Conference on AI and Interactive Digital Entertainment*.

Ware, S. G.; and Siler, C. 2021. Sabre: A Narrative Planner Supporting Intention and Deep Theory of Mind. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 17(1): 99–106.

Yao, L.; Peng, N.; Weischedel, R.; Knight, K.; Zhao, D.; and Yan, R. 2019. Plan-and-Write: Towards Better Automatic Storytelling. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01).

Young, R. M. 1999. Notes on the use of plan structures in the creation of interactive plot. In Mateas, M.; and Sengers, P., eds., *Narrative Intelligence: Papers from the AAAIFall Symposium (Technical Report FS-99-01)*.

Appendix A

A.1 Story Samples

The stories are generated backwards from the last sentence, which is provided by prompting GPT-J on ROCStory titles.

John took a train to get to home.
John connects internet to network cable at home
John used the internet to get to internet.
John bought lot of computer game to the internet at internet

John filled the car with gas on the way to the garage.
John drove to get to garage.
John took home his brown car for a treat at garage

John walked to get to park.
John buys bird watching lens from shop.
John loves to watch birds at park
John took a taxi to get to bank.
John exchanged the money at the bank.
John went to get to park.
John bought a bird from the pet shop.
John found a lost bird in the park at park

John took a bus to get to school.
John buys a pen and paper from school.
John wrote a school paper at school

John walked to get to office.
John attends a staff meeting at the office.
John attended a staff meeting at office

John took a bus to get to school.
John hires party at school
John hold a party in the class at school

John walked to get to shop.
John buys food at shop.
John managed to consume two more glasses of water at shop

John drove to get to police station.
John takes delivery case from the delivery van.
John delivered the case to the police at police station

John walked to get to computer shop.
John bought a new computer at computer shop

John took a taxi to get to school.
John played a prank on his student at school

John walked to get to shop.
John used his money to buy stuff at shop

John he got a phone call.
John took a taxi to get to home.
John gets a phone from the table.
Chris took a taxi to get to home.
John he had a call from Chris.
John went to get to department store.
John search for the stolen gifts at department store

John walked to get to store.
John buys a banjo from a store.
John walked home to get to home.
John played a song on his banjo at home

John drove to get to traffic jam.
John drive safely through the traffic jam at traffic jam

John his patient has heart pain.
John was taken to get to hospital.
John got stethoscope from medical kit.
John diagnosed heart pain in his patient at hospital

John walked to get to nursery.
John got baby from mother.
John pat the baby in the mother at nursery

John took a walk to get to bedroom.
John caught bugs in his bed at bedroom

John gets the car keys from the car.
John drove to get to car.
John fixes car light by car light at car
John check the airbag light in his car at car

John drove to get to park.
John played a game with friends at park

John walked to get to sword shop.
John buys a sword from a sword shop.
John drove to get to park.
John beat his neighbor in a duel at park

John took a bus to get to home.
John makes party plan at home
John and his friends plan a party at home

Liseh walked to get to store.
Liseh gets a phone from a store.
John took a bus to get to office.
Liseh took a bus to get to home.
Liseh gets help from John.

John loaded the money on the bus.
John took a bus to get to cinema.
John never met his dream girl at cinema

John took a bus to get to garden.
John got bitten by a squirrel in his garden at garden

John climbed down the ladder to get to home.
John gets ladder from the closet.
John took a ladder to get to tower.
John climb to the top of the tower at tower

John took a bus to get to restaurant.
John ate some pizza for breakfast at restaurant

John took a bus to get to school.
John asked for pencil from teacher.
John asked when he could have the next turn at school

John get business card from the card board.
John walked to get to board room.
John he has to discuss about important business at board room
John has booking from hotel service.
John took a taxi to get to hotel.
John has meeting room booked from hotel service.
John went to get to board room.
John has to have the meeting at board room
John took a taxi to get to office.
John had a meeting with the boss at office
John went to get to board room.
John gets appointment letter from secretary.
John get board room by appointment at board room
John walked to get to cleaning closet.
John gets cleaning cloth from the cleaning closet.
John went to get to board room.
John cleaned the board room at board room
John discuss the important matter in the board room at board room

---

John he lost his glasses.
John went to get to optician.
John his old glasses broke at optician
John bought the new glasses at optician

John drove to get to office.
John he has to work late at office
John he has to work overtime at office
John has to work late at office
John walked home to get to home.
John cannot get a mosquito net at home
John mosquitos chased him at night.
John gets mosquito net from the basket.

John ran to get to forest.
John escaped from the mosquitos.

John drove to get to car.
John drove his car at car

John walked to get to beach.
John saw the sunset and the rainbow at beach

John drove home to get to home.
John pulled the son out of the fire at home

John walked to get to playground.
John skipped rope for a long time at playground

John drove to get to garage.
Jack garage wasn't in his neighborhood.
Jack took a cab to get to garage.
John gave jack a check at garage
John gave jack his car service at garage

John walked to get to kitchen.
John gets a kitchen knife from the kitchen.
John cooks food at kitchen
John walked home to get to home.
John he can feed the pugsy at home
John loved pets at home
John fed the pugsy at home

John drove to get to team building.
John bought a ticket for team building event at team building
John participate in team building at team building

John drove to get to post office.
John didn't feel good about it at post office
John he it didn't like the post at post office
John it made him feel good and enjoy at post office
John enjoyed with the post of amusing mail at post office

John got flu shot from nurse.
John took a taxi to get to hospital.
John's she has the flu at hospital
John John's girlfriend don't feel well.
John walked to get to library.
John lost his girlfriend at library
John bought a book from the library.
John searched for the reason of his broken heart at library

John walked to get to cinema.
John he didn't like the film at cinema
John watched nothing at cinema

John walked to get to social media.
John posted rude comments about his friend at social media
John had too much trouble in social media at social media

John walked to get to cinema.
John buys a movie ticket from ticket office.
John watched the movie at cinema

John walked to get to ticket office.
John puts the money in the ticket at ticket office
John walked to get to cinema.
John buys a ticket from the ticket office.
John walked to get to concert.
John became bored at the concert.

John his colleagues didn't like him.
John rode a taxi to get to taxi.
John counted the money in a taxi at taxi
John took a taxi to get to office.
John didn't get any raise at office
John doesn't like his colleagues at office
John ordered and finished the slow work day at office

John took a bus to get to kitchen.
John gets a spoon from the kitchen.
John swallow the sugar at kitchen

John walked to get to store.
John buys skates from the store.
John walked to get to kitchen.
John sharpened his skates at the kitchen.
John walked to get to lake.
John ice skated on the frozen lake at lake

John took a taxi to get to party.
Julie drove to get to party.
John and Julie met at a party.

John walked to get to phone booth.
John buys a phone from the phone booth.
John took a taxi to get to home.
John used the phone to call his friend at home

John his smoke alarm went off.
John took a car to get to home.
John he feared for his life at home
John fled the smoke alarm at home

John walked to get to near the river.
John had never had a pet at near the river
John walked to get to neighbour.
John bought a puppy from the next door neighbour.

A.2. Few-shot Prompt for The GPT-J-6B Baseline

Here is an story on the topic ”Drained Battery”.
Tom let his friend borrow his phone. The friend kept using it. The friend kept draining the battery. Tom got it back way later. The phone died shortly after.

Here is an story on the topic ”Matthew Makes Good And Does Good”.
Matthew would spend hours working on his pitching skills. By high school, Matthew was being scouted by the big leagues. Matthew became a famous pitcher and had a long career. Matthew’s only regret was never having kids. Matthew started a charity to buy sports equipment for poor kids.

Here is an story on the topic ”Race”.
Ray was the slowest on the team. Ray needed a 6 minute mile in order to qualify for the race. Ray decided to train an extra hour everyday. Ray eventually got a 6 minute mile. Ray was able to join the race and he did very well.

Here is an story on the topic ”A Constant Struggle”.
Jeff had always wanted to be a pilot. Jeff would spend hours on flight sims in his garage. Jeff found a school that would teach him how to fly. Jeff signed up to learn to fly a plane that day. Jeff eventually became a pilot for a major airline.
Here is an story on the topic ”$[title]$”.