NEURAL STORY PLANNING

Authors: Anbang Ye, Christopher Cui, Taiwei Shi, and Mark O. Riedl
Georgia Institute of Technology
{aye42, ccui46, maksimstw, riedl}@gatech.edu

27th International Joint Conference on Artificial Intelligence (IJCAI) 2018

Presenter: Shawn Bray

AGENDA:

- Abstract
- Introduction
- Related Works
- Neural Story Planner
  - Precondition generation
  - Event generation
  - Final total ordering
- Algorithm
- Evaluations
- Limitations and weakness
- Conclusion
  - Relate to story generation and interactive fiction
  - Which part is specifically for story generation
  - Which part is specifically for interactive fiction

ABSTRACT:

The challenge lies in creating an automated plot involves crafting a sequence of events that make a coherent story. Traditional planners start from the end goal but rely on predefined actions. Neural language models can generate diverse stories but struggle with coherent endings.

The authors introduces a method that combines **causal planning** with **neural language models**. The proposed system identifies preconditions for events and then finds events that will cause those conditions to come true.

INTRODUCTION:

Early methods for story generation relied on **symbolic planning** and **case-based reasoning**. Symbolic planners ensure the logical soundness of a plan and enhance character believability, conflict, and some basic theory.

However, symbolic planners need hand-crafted schemas to define preconditions and effects. This combination of methods have shown to have **limited diversity**.

Neural language model offer a solution which can generate **diverse stories**.

This new combination, using a pre-trained language model to deduce events and their preconditions, by working backward from the story's ending has shown positive results.

RELATED WORKS:

Previous methods for story generation produce coherent plotlines but often lack diversity and length. They were also not aware of story's ending, which made them less realistic.

There have been some methods that try to address these limitations:

- **EDGAR** – generates stories backward using LLMs based on question answering.
  Lack of natural flow in narrative.
- **C2PO** - Conducts bidirectional search using the COMET model for commonsense inference.
  Huge overhead cost and maintaining was very complex.
- **TattleTale** - Uses a classical symbolic planner to sequence actions and condition language model.
  Needs significant fine tuning.

These models typically do not have a concept of the story's ending, and lead to incoherent conclusions.

NEURAL STORY PLANNER

The proposed method generates causally coherent story plans using causal links from POCL plans.

The process begins with the final event and works backwards. Events and their preconditions are deduced by an LLMs. The planner starts with an input sentence, specifying the story's ending and a set of initial conditions. It then uses the LLMs to recursively identify the preconditions for any event in the plot.

For each precondition, the proposed system picks the first valid event concluded. To help reduce the logical impossibilities an open world setting is used. If the inference leads to a repeated pattern of events, then the planner backtracks and selects an alternative route.

Partial Order Causal Link Planning(POCL) like a TO-DO list where task can be done in different orders.

NEURAL STORY PLANNER (PRECONDITION GENERATION 1)

For the precondition generation, the proposal uses LLMs to figure out what need to happen before an event occurs.

Then, the prompts are used to guild questions for the LLMs. These prompts are crafted based on how humans naturally understand stories.

Next, it classifies these preconditions into six types:

Item Need: what a character needs to perform an action.

Item State: what conditions must be true about an item.

How: what previous events made the current action possible.

Interactions with Others: interactions between characters needed for an action.

Reason: Identifies the motivation for an event.

Location: the required setting for each action.

NEURAL STORY PLANNER (PRECONDITION GENERATION 2)

Next it removes any unwanted preconditions first, based on wants and needs.

Then, it ensures the correct words formed are chosen next. By using **Pointwise Mutual Information with Domain Constraints** to evaluate and choose the best word form for the context.

Also, **Cosine Similarity** is used to filter out redundant words to maintain diversity and clarity.

NEURAL STORY PLANNER (FINAL TOTAL ORDERING 1)

Develop a complete list of events in the plot.

It then prioritize certain sequences over others to ensure coherence.

It then organizes the events based on the type of precondition they satisfy:

- Events that initiate character actions come first.
- Events that build on already established elements come next.

This ensures that the plot flows logically and maintains a strong narrative structure.

ALGORITHM

- **Initialize:** Start with the end of the story. You have an input sentence specifying the story's ending and initial conditions.
- **Backward Search:**
- Use a LLM to recursively conclude preconditions for the final event.
- For each concluded precondition, semi-greedy selection of the first event that meets the precondition is done.
- **Event Generation:**
- If an event is concluded, it generates the necessary preceding events and their preconditions.
- Continues this process, creating a chain of events leading up to the final event.

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-f 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 precondit
    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 precondit
    else
      $event' \leftarrow generate\ event(c, context)$
      $P \leftarrow P \cup \{event' \xrightarrow{c} event\}$ ▷ Satisfy with new ev
      $queue \leftarrow queue \cup \{event'\}$

Figure(1): Neural Plot Planner algorithm [1]

ALGORITHM

- **Heuristics and Constraints:**
- Uses heuristics to decide when a precondition type is unnecessary.
- Avoids logical impossibilities using an open-world setting.
- **Cycle Detection and Backtracking:**
- If an inference leads to a repeated event, the system backtracks.
- Selects alternative inferences to ensure a coherent, non-repetitive plot.
- **Precondition Satisfaction:**
- Ensures all preconditions of the events are satisfied by previously generated events.
- Uses causal links to maintain logical connections between events.
- **Output:**
- The final output is a causally coherent sequence of events, starting from the initial conditions and leading to the story's conclusion.

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-f 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 precondit
    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 precondit
    else
      $event' \leftarrow generate\ event(c, context)$
      $P \leftarrow P \cup \{event' \xrightarrow{c} event\}$ ▷ Satisfy with new ev
      $queue \leftarrow queue \cup \{event'\}$

Figure(1): Neural Plot Planner algorithm [1]

EVALUATIONS:

Coherence Measure

- Overall Accuracy: 78.20%
- ROCStories response rate of 85.39%

- Validate measure with 100 stories from ROCStories dataset, half of which should be answerable, and half should produce "none"
- Measure has overall accuracy of 78.20%

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.

GPT-J-6B: Plot-generating large model
C2PO: Commonsense, interpolates story events
comGen: Transformer, fine-tuned, commonsense
plan-write-revise: Fine-tuned, title-based storytelling

Rochester Human-Computer Interaction Stories (ROCStories) – dataset to help evaluate story generation. How well system handles this datasets story generation task.

WEAKNESS:

Not all events need all six types of preconditions. Forcing the language model to generate unnecessary preconditions can lead to unpredictable outcomes.

Also, this method uses heuristics to decide when a precondition type is not needed, which can sometimes be inaccurate.

Finally, the system's performance is constrained by the capabilities of the language model, limiting its ability to create highly complex or imaginative plots.

LIMITATIONS:

- **Advantages:**

1. **Causal Coherence:** Ensures logical, engaging stories.
2. **Dynamic Adaptation:** Generates unique storylines.
3. **Depth & Complexity:** Creates rich, layered plots.

- **Limitations:**
- **Heuristic dependency:** could limit creativity and autonomy.
- **Model Constraints:** affects quality and coherence.
- **Cycle Detection:** backtracking can be computationally intensive.

How does this relate to story generation and interactive fiction

- Narrative Coherence: Ensures a logical and immersive story flow.
- Dynamic Adaptation: Generates events that change based on player choices, creating unique storylines in interactive fiction.
- Plot Complexity: guides the development of detailed, multi-layered narratives.
- Responsive Content: Adjusts to player actions in real-time.

Which part is specifically for story generation

- Precondition Generation: It uses LLMs to conclude what must be true for events to happen in a story.
- Event Generation: creates coherent sequences and causal links in the narrative.
- Final Total Ordering: Explains how to order events to create a logical and engaging storyline, considering character intentions and preconditions.

Which part is specifically for interactive fiction

- Dynamic Storylines: It generate events based on player choices, creating unique and personalized narratives.
- Adaptive Content: The methods use of preconditions and event generation to adapt the story to player actions in real-time.

References:

[1] Ye, Anbang, Christopher Cui, Taiwei Shi, and Mark O. Riedl. "Neural Story Planning." Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI), 2018, pp. 5295–5299.

QUESTIONS??