Playing the Book: An Industrial Campaign Engine for Adversarial Long-Form Story Generation

Shyamal Chandra
Crankshaft News

2026

Abstract

Newsrooms and story studios now ask language models for novel-length drafts, then discover that a single context window cannot hold a 500-page plot, cannot keep secrets, and cannot stay surprising without becoming incoherent. We report an industrial system, the *adversarial story generator*, that treats a manuscript as the export of a played campaign rather than as one generation. A Rust command-line service drives local or cloud Ollama models on two timescales—strategy and tactics—persists world state in SQLite, races parallel writer drafts, and publishes a `book`-class TEX file compiled with `pdflatex`. The only required human input is one opening sentence. We describe the product form (an *adversarial labyrinth*), the closed agent loop, operational constraints from a 500-page batch, and lessons from shipping the engine as a private Crankshaft News tool.

1 Industrial problem

A 500-page book is on the order of 125,000 words. Production models, even with large windows, do not reliably maintain faction goals, planted objects, and delayed payoffs across that span if the only memory is the transcript. Industry practice of "just prompt it again" produces three failure modes we observed in newsroom pilots:

1. **Context collapse.** Mid-book chapters forget the opening line or leak a twist that was scheduled for the last act.
2. **Unfair surprise.** The model invents a reversal that no earlier scene could have supported.
3. **Job fragility.** A single HTTP timeout or a picky critic aborts an overnight batch and wastes compute.

Crankshaft News needed a *batch product*: an operator types one line ("The rookie from Pittsburgh, PA cannot run but can coach."), leaves the machine, and returns to a typeset PDF—or resumes from disk after a crash. The system had to run on a laptop against `ollama serve` and, when needed, against Ollama Cloud with an API key. It had to be a CLI, not a web app, and the published artifact had to be TEX plus `pdflatex`, matching existing print workflows.

2 Product form

We name the target form an *adversarial labyrinth*: long-form narrative whose hidden state is designed to invalidate confident reader predictions while remaining causally fair. Related labels used internally are *peripeteic campaign* (reversal as the unit of play) and *narrative wargame* (factions, fog of war, victory conditions). The manuscript is the after-action report.

This is not interactive fiction in v1. The loop is autonomous. Genre, cast, and premise are inferred from the opening line; we deliberately refuse a questionnaire so operators can fire-and-forget.

3 System architecture

Truth lives in Rust structures persisted to SQLite (WAL), not in model prose. Table 1 lists the five bounded JSON roles. Each call receives a short system prompt, a *slice* of world state (unlocked facts, recent scenes, chapter rollup), and a schema. Locked secret *content* is not shown to the critic; a Rust scan is the hard leak gate.

Table 1: Campaign roles. Strategy uses lower temperature; tactics use higher temperature.

Role	Horizon	Responsibility
Strategist	slow	Title, factions, secrets, twist calendar, chapter objectives
Tactician	fast	One scene beat: goal, obstacle, information delta
Adversary	fast	Optional beat mutation when the implied reader is too sure
Writer	fast	400–800 words of plain prose (never LaTeX)
Critic	fast	Continuity only; Rust owns secret leakage

A 500-page target is shaped as $\lceil P/10 \rceil$ chapters and five scenes per chapter when $P \geq 20$ (fifty by five at $P = 500$). Chapter 1 cannot spend an unearned campaign reversal. After each chapter, a $\sim$400-token rollup is the only long-term prose memory.

3.1 Publication path

Agents never emit LaTeX. Rust escapes specials, maps a few Unicode punctuation marks for Latin-1 `pdflatex`, and wraps `book` with `fontenc`, `inputenc`, `microtype`, `setspace`, and `geometry`. Operators choose body font, point size, and `-single-lined` compact pages. Compilation is `pdflatex` twice; a failed compile keeps the `.tex` and exits non-zero.

4 Implementation notes

The crate `adversarial-story` is a library plus CLI (`tokio` multi-thread, `reqwest`, `rusqlite`, `clap`, `indicatif`). Ollama is called at `/api/chat` in JSON mode. `-source local` defaults to `http://127.0.0.1:11434` with no key. `-source cloud` defaults to `https://ollama.com` and requires `OLLAMA_API_KEY`. `-m`/`-model` sets both layers; `-strategy-model` and `-tactics-model`

override. `-j/-workers/-parallel` races writer/critic drafts; a semaphore caps concurrent HTTP.

Every accepted scene is a checkpoint: `next_chapter`, `next_scene`, and a WAL flush. `-continue` resumes without re-typing the opening line if `-tex` or `-db` is given. `-restart` wipes the job. A progress bar reports chapter, scene, phase, and worker count.

The opening line is accepted as a positional argument, `-opening`, stdin (`-`), or a TTY prompt. Empty starters are rejected. That line is committed as scene 0 of chapter 1 and is the first body paragraph of the book.

5 Operational lessons

Do not let the critic own the batch. An early production run died on chapter 1, scene 1 with "Unearned Reversal; Leaked Secrets" after the Rust leak scan had already passed. The model had been shown locked secret text in its prompt and then "detected" leaks. We now ignore hallucinated leak vetoes, strip chapter-1 reversals, and commit the last leak-free draft after retries.

Parallelism is for drafts, not chapters. Scene n mutates world state for scene $n + 1$. We parallelize writer candidates inside a scene, not chapters across the book.

Checkpoints are the product. A 500-page local batch is hours of tokens. Resume from SQLite is mandatory; re-prompting the whole book is not a recovery plan.

TEX is a compiler, not a model skill. Asking a chat model for a 500-page `.tex` file fails. Escaping in Rust is cheaper and testable.

6 Limitations and next work

v1 has no human-in-the-loop move, no embeddings recall of planted objects, no XeLaTeX/LuaLaTeX, and no TUI. Mid-book "replan" is the next chapter's objectives plus a rollup, not a full rewrite of the twist ledger. Cloud model names are operator-selected; `-list-models` prints `/api/tags` when the backend supports it.

7 Conclusion

Long adversarial fiction is a *game* with two clocks and a disk. The industrial contribution is not a new base model. It is a Rust campaign service that keeps secrets in SQLite, spends twists on a calendar, races drafts, and ships a typeset book from a single sentence. That is how we generate a 500-page labyrinth without asking one prompt to remember the whole maze.

Acknowledgments

This work was done at Crankshaft News. The engine is private software.

Copyright (C) Shyamal Chandra, Crankshaft News, 2026.