

## **Agentic AI & Context Management Architecture**

This section details the core innovation of the project: the **Context-Snoopiest** architecture. This system manages the trade-off between the infinite depth of a novel and the finite constraints of the LLM context window.

### **The Problem: Why Standard RAG Fails for Narrative**

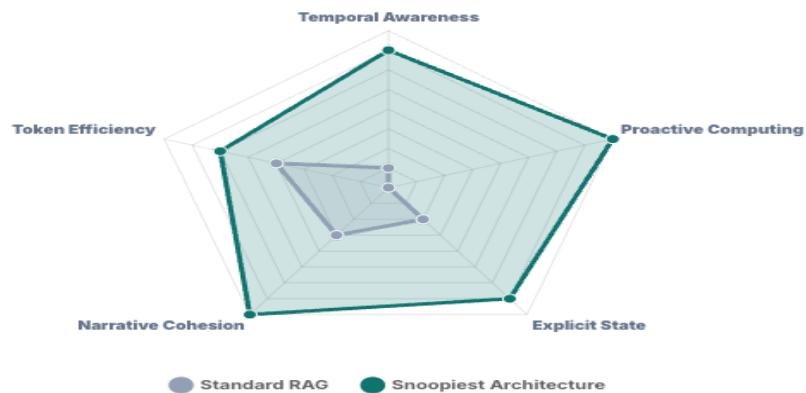
Retrieval-Augmented Generation (RAG) is designed for fact retrieval, not causal narrative logic. A novel is a chain of events where “State” (who has what, who is where) matters more than semantic similarity.

# Architecture Profile

While standard Context Management treats data spatially (finding "nearby" vectors), **Snoopiest** treats data chronologically and structurally. This results in a massive shift towards explicit state management and temporal awareness.

## ● STANDARD CODE RAG

## ● CONTEXT-SNOOPIEST



## PRIMARY MECHANISM

### RAG & Sliding Window

Indexes vectors; retrieves based on similarity match.

### Hierarchical Summarization

Maintains a recursive tree of narrative summaries (Level 1-3).

## TEMPORAL AWARENESS

### Low (Spatial)

Treats code/text as a dependency graph or proximity vector.

### High (Chronological)

Treats text as a causal sequence (Cause → Effect → Outcome).

## CONTEXT RETENTION

### Reactive

Retrieves data only after a user query is made.

### Proactive

Pre-computes context ("Backbone") before processing chunks.

## STATE MANAGEMENT

### Implicit

Relies on raw file content and git history diffs.

### Explicit DB

Uses JSON "Character Sheets" & "Inventory" as state databases.

## SUMMARIZATION STRATEGY

### Compaction

Compresses history primarily to fit token limits.

### Narrative Synthesis

Rewrites beats into higher-level abstractions preserving meaning.

## HANDLING OVERFLOWS

### Truncation

Drops oldest turns or summarizes purely by count.

### Critical Locking

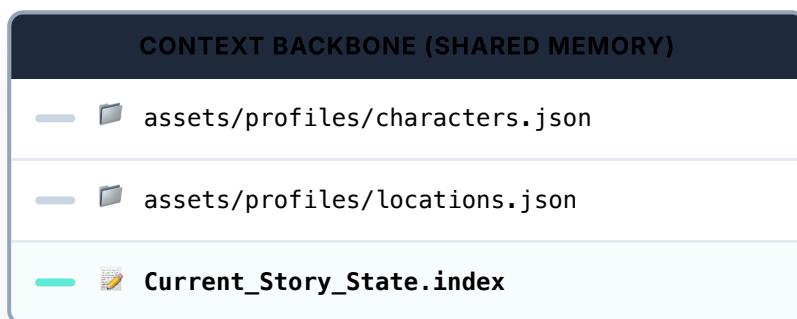
Drops non-essentials while "Locking" critical plot points.

## The “Snoopiest” Hierarchical Strategy

To solve the narrative decay problem, we implement a recursive, tree-structured memory system that ensures the agent never loses the “thread” of the story.

## THE CONVEYOR BELT

Snoopiest Context Management Metaphor



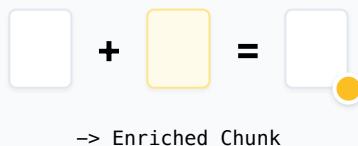
A

CONTEXT INJECTOR

### The "Stapler"

Receives raw text chunk. Reaches into the cabinet to find matching **Character Profiles**.

**ACTION:**



B

THE DIRECTOR

## Translation Bot

Reads the enriched chunk. Ignores narrative fluff. Writes technical **Camera Directives**.

### OUTPUT:

```
{  
  "shot": "Medium",  
  "move": "Pan Right",  
  "focus": "Scar"  
}
```

C

THE ARCHIVIST

## Memory Keeper

Summarizes the chunk. Creates a new **Index Card** and files it back in the Cabinet.

### UPDATE:

Context Window Optimized



CONTEXT RETRIEVAL



VIDEO GENERATION



CONTEXT UPDATE

## Memory Tier Breakdown

### Level 0: The Working Window

The raw text of the current scene (approx. 2000 tokens). This is where the high-resolution action takes place.

## **Level 1: Scene Summaries**

As a scene completes, it is compressed into a dense factual summary (50-100 words), capturing state changes rather than prose.

## **Level 2: Chapter Synthesis**

Once a chapter is complete, Level 1 summaries are synthesized into a mid-term memory layer that removes transient details.

## **Level 3: The Narrative Backbone**

The “Long-Term Memory” layer. A continuously updated document tracking global arcs across the entire 100,000+ word novel.

---

## **Proactive Context Management**

Unlike systems that simply slide a window (dropping tokens by age), Context-Snoopiest utilizes **Semantic Retention**. The agent explicitly decides *what* to keep. If a vital plot point occurs on Page 1, it is “locked” into the Level 3 backbone for the duration of the project.