# Narrative Analysis System Design

## Overview

A comprehensive system for analyzing prose to discover narrative patterns, stylistic choices, and story structures. The goal is to identify "recipes" that make stories compelling and create a searchable database of narrative tropes and patterns.

## Core Components (1-4)

### 1. Text Ingestion Layer

**Purpose**: Convert raw text into structured, analyzable format

**Components**:

- **Format Parser**: Handle multiple file types (txt, epub, pdf, docx)
- **Text Cleaner**: Remove formatting artifacts, normalize encoding
- **Chunking System**:
  - Paragraph-level segmentation
  - Scene/chapter detection
  - Dialogue vs. narrative separation
- **Metadata Extractor**: Author, publication date, genre tags
- **Text Indexer**: Create positional references for every word/sentence

**Output**: Structured document object with hierarchical text organization

### 2. Multi-Level Analysis Engine with Clue System

**Purpose**: Extract linguistic and literary features while constructing contextual clues for dynamic rule creation

**Core Innovation**: The **Clue-Based Rule Construction System** - instead of fixed grammar rules, the engine dynamically constructs parsing rules based on accumulated textual evidence.

#### 2.1 The Clue Collection Framework

**What is a Clue?** A clue is a contextual pointer that connects:

- Text location (sentence, paragraph, chapter)
- Linguistic evidence (syntax pattern, semantic content)
- Confidence score (how certain we are about this connection)
- Rule implications (what parsing decisions this supports)

**Clue Types**:

- **Syntactic Clues**: Unusual sentence structures, repeated patterns
- **Semantic Clues**: Character relationships, thematic connections
- **Narrative Clues**: Story function indicators, plot beats
- **Stylistic Clues**: Author voice markers, genre conventions
- **Contextual Clues**: State-dependent information (character mood, story tension)

**Clue Structure**:

```
Clue {
  source_location: TextPointer
  evidence_type: ClueType
  confidence: Float (0.0-1.0)
  related_locations: [TextPointer]
  rule_suggestion: ParseRule
  validation_status: Enum(Unverified, Confirmed, Contradicted)
}
```

### 2.2 Dynamic Rule Construction Engine

**The Revolutionary Concept**: Instead of parsing with fixed rules, we construct rules from accumulated clues.

**Rule Construction Process**:

1. **Collect Evidence**: Gather clues from multiple analysis passes
2. **Pattern Recognition**: Identify recurring clue combinations
3. **Rule Hypothesis**: Generate tentative parsing rules
4. **Cross-Validation**: Test rules against other text sections
5. **Rule Refinement**: Adjust rules based on success/failure patterns

**Example Rule Evolution**:

Initial Clue: "Character X mentioned in dialogue heavy section"
+ Supporting Clue: "Emotional language spike detected"
+ Pattern Clue: "Similar structure in chapter 3, 7, 12"
= Generated Rule: "When X appears in dialogue, expect emotional revelation pattern"

### 2.3 Context-Aware Syntactic Analysis

**Beyond Traditional Parsing**: Syntax rules that adapt based on narrative context

**Adaptive Grammar System**:

- **Base Grammar**: Standard English syntactic rules
- **Context Modifiers**: Rules that change based on story state
- **Style Adjustments**: Grammar variations for different authors/genres
- **Clue-Informed Parsing**: Ambiguity resolution using collected clues

**Context Types**:

- **Narrative Context**: Dialogue vs. description, action vs. reflection
- **Character Context**: Whose perspective, emotional state
- **Genre Context**: Literary fiction vs. genre conventions
- **Author Context**: Individual stylistic patterns

### 2.4 Semantic Analysis with Clue Integration

**Meaning Discovery Through Evidence Accumulation**

**Clue-Driven Semantic Analysis**:

- **Entity Recognition**: Characters, places tracked across entire text
- **Relationship Mapping**: Connections discovered through clue correlation
- **Thematic Extraction**: Recurring concepts identified via clue patterns
- **Emotional State Tracking**: Character/narrative mood through contextual clues

**The Breakthrough Insight**: Semantic meaning emerges from the interaction of clues rather than from individual word analysis.

### 2.5 Narrative Function Detection

**Story Beat Recognition Through Clue Synthesis**

**Function Detection Algorithm**:

1. **Collect Narrative Clues**: Plot markers, character actions, tension indicators
2. **Cross-Reference Patterns**: Compare with known story templates
3. **Generate Hypotheses**: Propose narrative function based on clue evidence
4. **Validate Through Context**: Check consistency with surrounding story elements

**Narrative Clue Examples**:

- **Setup Clues**: Character introduction patterns, world-building density
- **Conflict Clues**: Tension language, obstacle mentions, character opposition
- **Resolution Clues**: Conclusion language, character growth indicators

### 2.6 Stylistic Pattern Recognition

#### Author Voice Through Clue Accumulation

**Style Clues**:

- **Sentence Structure Patterns**: Length distributions, complexity variations
- **Word Choice Patterns**: Abstraction ladder preferences, register consistency
- **Rhythm Patterns**: Cadence, punctuation usage, paragraph flow
- **Rhetorical Patterns**: Metaphor frequency, dialogue tags, description style

### 2.7 The Clue Validation System

#### How to Know if Clues are Reliable

**Validation Methods**:

- **Cross-Reference Validation**: Same pattern appears in multiple locations
- **Consistency Checking**: Clues don't contradict each other
- **Template Matching**: Clues align with known narrative patterns
- **Statistical Significance**: Pattern occurs more than random chance
- **Contextual Coherence**: Clues make sense within story context

**Confidence Scoring**:

- High Confidence: Multiple supporting clues, consistent patterns
- Medium Confidence: Some supporting evidence, minor contradictions
- Low Confidence: Single source, potential false positive

### 2.8 The Multi-Pass Clue Construction Process

**How the System Builds Understanding Over Time**

**Pass 1 - Basic Clue Collection**:

- Identify obvious patterns (dialogue, paragraphs, chapters)
- Collect surface-level stylistic clues
- Mark ambiguous sections for deeper analysis

**Pass 2 - Relationship Clue Discovery**:

- Connect entities across text using Pass 1 clues
- Identify recurring patterns and structures
- Build initial rule hypotheses

**Pass 3 - Context Integration**:

- Apply discovered rules to analyze previously ambiguous sections
- Refine rule confidence based on success/failure
- Discover meta-patterns (patterns of patterns)

**Pass 4 - Pattern Synthesis**:

- Combine all clue types into comprehensive understanding
- Generate final rule set for this specific text
- Extract transferable patterns for pattern database

## 3. Relationship Mapping System

**Purpose**: Track connections and dependencies across the text

### 3.1 Pointer System

- **Coreference Resolution**: Pronouns to their referents
- **Anaphoric References**: Backward-pointing references
- **Cataphoric References**: Forward-pointing references
- **Cross-Reference Database**: Store all pointer relationships

### 3.2 Character State Tracking

- **Character Presence**: Where each character appears
- **Emotional State Evolution**: Track character mood changes
- **Relationship Dynamics**: Character interactions over time
- **Knowledge State**: What each character knows when

### 3.3 Plot Progression Mapping

- **Causal Chains**: Event A leads to Event B
- **Temporal Ordering**: Chronological vs. narrative sequence
- **Subplot Tracking**: Multiple story threads
- **Foreshadowing Links**: Setup to payoff connections

### 3.4 Thematic Relationship Network

- **Motif Repetition**: Recurring symbols/concepts
- **Parallel Structures**: Similar scenes/situations
- **Contrast Patterns**: Opposing elements
- **Echo Detection**: Callback references

## 4. Pattern Recognition Engine

**Purpose**: Discover and catalog narrative "recipes" and stylistic patterns

### 4.1 Template Library (Story Recipes)

**Heist Template Example**:

- Inciting incident detection
- Team assembly sequence identification
- Preparation montage recognition
- Complication/twist markers
- False failure patterns
- Hidden plan revelation beats
- Resolution structure

**Other Template Categories**:

- **Quest Narratives**: Hero's journey variants
- **Romance Arcs**: Meet-cute to resolution patterns
- **Mystery Structures**: Clue placement and revelation timing
- **Thriller Pacing**: Tension build and release cycles
- **Comedy Beats**: Setup-punchline patterns

### 4.2 Pattern Matching Algorithms

- **Structural Similarity**: Compare narrative sequences
- **Functional Equivalence**: Different surface, same purpose
- **Template Scoring**: How well does text match known patterns
- **Variant Detection**: Identify pattern modifications

### 4.3 Trope Database

**Character Tropes**:

- Mentor figures and their fate patterns
- Reluctant hero characteristics
- Comic relief positioning and function
- Antagonist motivation patterns

**Plot Tropes**:

- Red herring placement strategies
- Chekhov's gun implementations
- Deus ex machina detection
- Plot armor identification

**Setting Tropes**:

- Atmospheric description patterns
- World-building information delivery
- Environmental storytelling techniques

### 4.4 Statistical Validation System

- **Frequency Analysis**: How often does pattern occur
- **Author Specificity**: Pattern unique to author vs. genre
- **Cross-Work Validation**: Pattern consistency across multiple texts
- **Significance Testing**: Statistical relevance of patterns

### 4.5 Search and Discovery Features

- **Pattern Search**: Find all instances of specific narrative structure
- **Similarity Search**: Find texts with similar patterns
- **Anomaly Detection**: Identify unusual or unique patterns
- **Comparative Analysis**: Compare pattern usage across authors/genres

### 4.6 Advanced Pattern Ideas

**Micro-Patterns**:

- Dialogue tag preferences ("he said" vs. "he whispered")
- Transition phrase usage ("Meanwhile" vs. "Later")
- Description ordering (appearance before personality vs. reverse)

**Macro-Patterns**:

- Chapter ending hooks
- Scene transition techniques
- Information revelation pacing
- Character introduction methods

**Meta-Patterns**:

- Genre convention adherence vs. subversion
- Reader expectation management
- Narrative voice consistency
- Time manipulation techniques

**Structural Patterns**:

- Parallel storyline convergence
- Nested narrative structures
- Frame story implementations
- Multiple perspective orchestration

## Data Structures

- **Document Tree**: Hierarchical text representation
- **Relationship Graph**: Network of all connections
- **Pattern Index**: Searchable pattern database
- **Feature Vectors**: Numerical representations for comparison
- **Timeline Objects**: Chronological event ordering

## Processing Pipeline

1. **Ingest** → Clean and structure raw text
2. **Parse** → Extract linguistic features at all levels
3. **Map** → Identify relationships and connections
4. **Recognize** → Match against known patterns and discover new ones
5. **Index** → Make everything searchable and comparable

## Success Metrics

- Can identify known story structures (like Ocean's Eleven pattern)
- Can distinguish between different authors' styles
- Can answer complex questions about narrative techniques
- Can discover previously unrecognized patterns
- Can predict reader engagement based on pattern analysis