Supplementary Materials:
Comparing Teaching Strategies through Misconception Analysis: A Case-Based AI Framework Using the EEDI Dataset

Dietmar Janetzko   Horacio González-Vélez
National College of Ireland

Overview

This document provides supplementary implementation details for the paper "Comparing Teaching Strategies through Misconception Analysis: A Case-Based AI Framework Using the EEDI Dataset." It includes:

1. Comprehensive algorithms for all five teaching personas
2. System architecture baseline algorithms
3. Implementation specifications
4. Prompt templates and examples

Contents

1 Terminology Framework

1.1 System Architecture Baselines

- **Traditional Baseline**: Standard teaching without CBR-AI system (M-score: 0.6802 ± 0.0084)
- **Pure CBR Baseline**: Retrieval-only system without AI generation (M-score: 0.6273 ± 0.0073, 7.8% improvement over Traditional)
- **Pure AI Baseline**: AI generation without case-based grounding (M-score: 0.6062 ± 0.0091, 10.9% improvement over Traditional)
- **Hybrid + Mnemonic**: Full system with mnemonic augmentation (M-score: 0.5840 ± 0.0068, 14.1% improvement over Traditional, 3.7% over Pure AI)

Note: M-score represents misconception persistence. Lower values indicate better performance (more effective remediation).

1.2 Teaching Personas

- **Socratic**: Question-driven guided discovery
- **Constructive**: Scaffolded knowledge building
- **Experiential**: Real-world contextualization

- **Rule-based**: Explicit procedural instruction
- **Traditional teaching**: Direct instruction approach

2 Teaching Persona Algorithms

2.1 Socratic Persona

The Socratic persona employs guided questioning to stimulate critical thinking through cognitive conflict.

Algorithm 1 Socratic Teaching Persona

```
Input: Student misconception M, student context C, retrieved cases R
Output: Sequence of Socratic questions Q

// Step 1: Analyze misconception pattern
pattern ← ExtractPattern(M)
contradictions ← IdentifyContradictions(pattern, R)

// Step 2: Generate counterexample
counterexample ← CreateCounterexample(M, C)
                    ▷ Select simpler case where misconception leads to obvious error

// Step 3: Construct question sequence
Q ← [ ]
Q.append("What happens if we try counterexample?")

// Step 4: Wait for student response and analyze
response ← StudentResponse()
recognition ← CheckRecognition(response, contradictions)

if recognition = True then
    Q.append("What does this tell us about our method?")
else
    Q.append("Can you explain why you got that result?")
end if

// Step 5: Guide toward generalization
if student understands flaw then
    Q.append("Can you apply this insight to the original problem?")
else
    return to Step 2 with simpler counterexample
end if

// Step 6: Validation
Q.append("Let's verify: what is the correct approach?")

Constraints:
- Never provide direct answers
- Maximum 5 questions per interaction
- Questions build on student responses
- Maintain encouraging tone

return Q
```

Theoretical Grounding: Based on Socratic method and cognitive conflict theory (Yang, 2005; Lipman, 2003).

Key Characteristics:

- Question-to-statement ratio: ≥ 0.8
- Average questions per interaction: 4.2
- Success rate: 71.8%

2.2 Constructive Persona

The constructive persona implements dynamic scaffolding based on Vygotsky's Zone of Proximal Development.

Algorithm 2 Constructive Teaching Persona

Input: Student misconception M, student context C, performance history H
Output: Scaffolded intervention I

// Step 1: Assess current ZPD level
$accuracy \leftarrow$ ComputeAccuracy(H) ▷ Weight: 0.50
$efficiency \leftarrow$ ComputeEfficiency(H) ▷ Weight: 0.30
$confidence \leftarrow$ ComputeConfidence(C) ▷ Weight: 0.20

$ZPD_{level} \leftarrow 0.50 \cdot accuracy + 0.30 \cdot efficiency + 0.20 \cdot confidence$

// Step 2: Determine difficulty level
$current_{difficulty} \leftarrow$ GetCurrentDifficulty(M)

if $ZPD_{level} > 0.75$ **then**
> $target_{difficulty} \leftarrow current_{difficulty} + 1$ ▷ Increase challenge
> $scaffolding_{level} \leftarrow$ Low

else if $ZPD_{level} < 0.45$ **then**
> $target_{difficulty} \leftarrow current_{difficulty} - 1$ ▷ Reduce complexity
> $scaffolding_{level} \leftarrow$ High

else
> $target_{difficulty} \leftarrow current_{difficulty}$
> $scaffolding_{level} \leftarrow$ Moderate

end if

// Step 3: Retrieve prerequisites
$prerequisites \leftarrow$ IdentifyPrerequisites(M)
$missing \leftarrow$ CheckMissingPrereqs($prerequisites, H$)

if $missing \neq \emptyset$ **then**
> // Address prerequisite gaps first
> **for** $prereq \in missing$ **do**
>> AddScaffolding(I, $prereq$, $scaffolding_{level}$)
>
> **end for**

end if

// Step 4: Build conceptual understanding progressively
$concepts \leftarrow$ DecomposeIntoConcepts(M)
SortByComplexity($concepts$)

for $concept \in concepts$ **do**
> $explanation \leftarrow$ GenerateExplanation($concept$, $scaffolding_{level}$)
> $example \leftarrow$ CreateWorkExample($concept$, $target_{difficulty}$)
> $practice \leftarrow$ CreatePracticeItem($concept$, $target_{difficulty}$)
>
> I.append($explanation$)
> I.append($example$)
> I.append($practice$)
>
> // Check understanding before proceeding
> $understanding \leftarrow$ AssessUnderstanding()
> **if** $understanding < 0.6$ **then**
>> $scaffolding_{level} \leftarrow$ Increase

Theoretical Grounding: Vygotsky's ZPD theory (Vygotsky, 1978) and scaffolding (Wood et al., 1976).

Key Characteristics:

- Adaptive difficulty adjustment: continuous
- Scaffolding levels: 3 (low, moderate, high)
- Success rate: 70.2%

2.3 Experiential Persona

The experiential persona contextualizes abstract concepts through real-world analogies.

Algorithm 3 Experiential Teaching Persona

```
Input: Student misconception M, student context C, context mappings CM
Output: Contextualized intervention I

// Step 1: Extract mathematical structure
structure ← ExtractMathStructure(M)
operations ← structure.operations
relationships ← structure.relationships

// Step 2: Search context network
candidates ← []
for context ∈ CM do
    sim ← StructuralAlignment(structure, context)
    if sim > 0.7 then
        candidates.append((context, sim))
    end if
end for

// Step 3: Select best context
SortByRelevance(candidates, C)                ▷ Consider student interests
best_context ← candidates[0]

// Step 4: Generate scenario
scenario ← CreateScenario(M, best_context)
scenario.setup ← "Imagine you're [context situation]..."

// Step 5: Map misconception to context
contextual_error ← MapMisconception(M, scenario)
scenario.problem ← "If we [apply misconception], what happens?"

// Step 6: Demonstrate consequence
consequence ← ShowConsequence(contextual_error)
scenario.outcome ← consequence

// Step 7: Reflect back to mathematics
reflection ← "Now, how does this relate to our fraction problem?"
scenario.reflection ← reflection

// Step 8: Construct complete intervention
I ← []
I.append(scenario.setup)
I.append(scenario.problem)
I.append(scenario.outcome)
I.append(scenario.reflection)

// Step 9: Provide abstract principle
principle ← ExtractPrinciple(M, scenario)
I.append("The key principle: principle")

// Step 10: Apply to original problem
I.append("Let's apply this principle to your original question...")
solution ← GuidedSolution(M, principle)
I.append(solution)
```

Theoretical Grounding: Kolb's experiential learning theory (Kolb, 1984) and situated cognition.

Key Characteristics:

- Context-concept mappings: 847
- Structural alignment threshold: 0.7
- Success rate: 76.3% (highest)

Example Context Mappings:

- Fraction addition → Recipe measurements
- Ratio/proportion → Map scales
- Percentage → Sales discounts
- Geometry → Architecture/design

2.4 Rule-based Persona

The rule-based persona follows systematic direct instruction principles.

Algorithm 4 Rule-based Teaching Persona

```
Input: Student misconception M, student context C
Output: Structured procedural intervention I

// Step 1: Review prerequisites
prerequisites ← GetPrerequisites(M)
I ← "Let's review what we need to know:"
for prereq ∈ prerequisites do
    I.append(ExplainBriefly(prereq))
    I.append(QuickCheck(prereq))                                   ▷ Verify understanding
end for

// Step 2: State the correct rule explicitly
rule ← GetCorrectRule(M)
I.append("The correct rule is: rule")

// Step 3: Break into small steps
steps ← DecomposeIntoSteps(rule)
I.append("Follow these steps:")
for i ← 1 to |steps| do
    I.append("Step i: steps[i]")
    if i < |steps| then
        I.append(CheckUnderstanding())                             ▷ Pause for verification
    end if
end for

// Step 4: Provide worked example
example ← CreateWorkedExample(M)
I.append("Worked Example:")
I.append(example.problem)

for i ← 1 to |steps| do
    I.append("Step i: steps[i]")
    I.append(ApplyStep(example, steps[i]))
    I.append("Result so far: " + ShowPartialSolution(example, i))
end for

I.append("Final answer: " + example.solution)

// Step 5: Identify the error in student's approach
student_approach ← ExtractApproach(M)
error_location ← FindError(student_approach, rule)
I.append("Your error: In your approach, at step error_location...")
I.append(ExplainError(error_location))

// Step 6: Provide corrective feedback
I.append("To fix this: rule[error_location]")

// Step 7: Guided practice
practice_problems ← GeneratePractice(M, difficulty=low, n=2)
I.append("Practice: Try these problems using the correct steps:")
for prob ∈ practice_problems do
    I.append(prob)
```

Theoretical Grounding: Rosenshine's principles of direct instruction (Rosenshine, 2012).
Key Characteristics:

- Step-by-step structure: rigid
- Worked examples: always included
- Success rate: 62.4%

2.5 Traditional Teaching Persona

The traditional teaching persona implements classical direct instruction methods.

Algorithm 5 Traditional Teaching Persona

```
Input: Student misconception M, student context C
Output: Direct instruction intervention I

// Step 1: Acknowledge the error
I ← "I see you got an incorrect answer. Let me explain the correct way."

// Step 2: Explain the concept broadly
concept ← GetUnderlyingConcept(M)
I.append("The concept here is: concept")
I.append(ProvideGeneralExplanation(concept))

// Step 3: Show the correct procedure
procedure ← GetStandardProcedure(concept)
I.append("Here's how to solve this type of problem:")
I.append(procedure)

// Step 4: Demonstrate with the problem
I.append("For your problem:")
solution ← SolveStep By Step(M)
for step ∈ solution.steps do
    I.append(step)
end for
I.append("Therefore, the answer is: solution.result")

// Step 5: Point out the error
I.append("Your mistake was: " + IdentifyError(M))

// Step 6: Provide practice suggestion
I.append("To master this, practice similar problems.")
I.append("Remember: KeyRule(concept)")

return I
```

Theoretical Grounding: Classical direct instruction pedagogy.
Key Characteristics:

- Structure: explain → demonstrate → practice
- Adaptivity: minimal
- Success rate: 65.7%

Distinction from Rule-based: While both use direct instruction, the traditional teaching persona is broader and less systematically procedural than the rule-based persona. Traditional teaching provides general explanations and demonstrations, whereas rule-based instruction follows Rosenshine's specific principles with explicit step-checking and structured feedback loops.

3 System Architecture Baseline Algorithms

3.1 Baseline (No CBR-AI System)

This represents standard teaching without computational support.

Algorithm 6 Baseline System (No CBR-AI)

```
Input: Student misconception M, student context C
Output: Standard teaching response R

// Simulate teacher's intuitive response
R ← "Let me explain this concept..."

// Provide general explanation based on teacher knowledge
explanation ← GeneralExplanation(M.topic)
R.append(explanation)

// Show example
example ← StandardExample(M.topic)
R.append(example)

// Suggest practice
R.append("Practice will help you master this.")

Characteristics:
- No case retrieval
- No AI generation
- No personalization
- Fixed intervention patterns

Performance:
- Baseline score: 0.658
- Improvement: 0% (reference point)

return R
```

3.2 Pure CBR Baseline

Case-based reasoning without AI generation.

Algorithm 7 Pure CBR Baseline

Input: Query case q, case base $\mathcal{CB}$, $k = 5$
Output: Retrieved intervention I

// Step 1: Extract features from query
$f_q \leftarrow [q.\text{topic\_complexity}, q.\text{prior\_performance},$
$\quad q.\text{misconception\_frequency}, q.\text{prerequisite\_count}]$

// Step 2: Compute similarities
$similarities \leftarrow [\,]$
for $c \in \mathcal{CB}$ **do**
  $f_c \leftarrow \text{ExtractFeatures}(c)$
  $sim \leftarrow 1 - \text{CosineSimilarity}(f_q, f_c)$
  $similarities.\text{append}((c, sim))$
end for

// Step 3: Sort and select top-k
Sort($similarities$, key=similarity, order=descending)
$top_k \leftarrow similarities[0 : k]$

// Step 4: Select best case
$best \leftarrow \text{Max}(top_k, \text{key}=c.\text{success\_rate})$
$I \leftarrow best.\text{intervention}$

Characteristics:
- Pure retrieval: no generation
- k-NN with cosine similarity
- Historical success weighting
- No LLM involvement

Performance:
- Score: 0.605
- Improvement: 8% over Baseline
- Literature: Kolodner (2014)

return I

3.3 Pure AI Baseline

AI generation without case-based grounding.

Algorithm 8 Pure AI Baseline

Input: Student misconception M, persona type P
Output: Generated intervention I

// Step 1: Validate persona
if $P \notin$ {socratic, constructive, experiential, rule_based, traditional_teaching} **then**
    $P \leftarrow$ socratic ▷ Default
end if

// Step 2: Build generic prompt (no cases)
prompt $\leftarrow$ "You are a P teaching assistant."
prompt.append("Student misconception: M.description")
prompt.append("Topic: M.topic")
prompt.append("Generate intervention using P approach.")

// Step 3: Generate with LLM
$I \leftarrow$ LLM.Generate(*prompt*, temperature=0.7, max_tokens=500)

Characteristics:
- No case retrieval
- Generic prompting
- LLM training data only
- No empirical grounding

Performance:
- Score: 0.586
- Improvement: 11% over Baseline
- Literature: VanLehn (2011)

return I

3.4 Hybrid + Mnemonic System

Full system with mnemonic-augmented CBR and AI generation.

Algorithm 9 Hybrid + Mnemonic System

```
Input: Query q, case base CB, LLM, persona type P
Output: Grounded intervention I

// Step 1: Mnemonic-enhanced retrieval
retrieved ← MnemonicRetrieval(q, CB)                ▷ Algorithm 1 from paper

// Mnemonic techniques applied:
// - Chunking: Hierarchical clustering into 10 clusters
// - Associative: Semantic network connections
// - Retrieval cues: Weighted features [0.35, 0.30, 0.15, 0.10, 0.10]
// - Elaboration: Prerequisites, reasoning patterns, connections

// Step 2: Format cases for prompt
case_examples ← [ ]
for c ∈ retrieved do
    example ← {
            description: c.misconception,
            strategy: c.intervention.strategy,
            outcome: c.outcome,
            context: c.elaborative_context
    }
    case_examples.append(example)
end for

// Step 3: Construct grounded prompt
prompt ← BuildHybridPrompt(P, q, case_examples)

// Prompt structure:
// [PERSONA_GUIDANCE]
// Historical Context: [RETRIEVED_CASES]
// Current Situation: [QUERY]
// Task: Generate grounded intervention

// Step 4: Generate with LLM
I ← LLM.Generate(prompt, temperature=0.7, max_tokens=300)

Characteristics:
- Mnemonic-enhanced retrieval
- Case-grounded generation
- Four cognitive techniques
- Persona-specific adaptation

Performance:
- Score: 0.559
- Improvement: 15% over Baseline
- Literature: This study

return I
```

4 Prompt Template Specifications

4.1 Socratic Persona Prompt Template

```
System: You are a Socratic teaching assistant. Your goal is to guide
students to discover correct understanding through carefully sequenced
questions that expose contradictions in their reasoning.

Historical Context - Similar Cases:
[CASE_1]: Students with [PATTERN_1] were guided to understanding through
counterexample [STRATEGY_1] (success: 89%)
- Key insight: [ELABORATION_1]

[CASE_2]: Students with [PATTERN_2] responded well to [STRATEGY_2]
- Related concepts: [ASSOCIATIONS_2]

Current Student:
Misconception: [MISCONCEPTION]
Student's reasoning: [REASONING]
Confidence: [CONFIDENCE]

Task: Generate 3-5 Socratic questions that:
1. Begin with a counterexample showing the flaw
2. Guide recognition of the contradiction
3. Lead toward correct generalization

Constraints:
- Never provide direct answers
- Build on student responses
- Maintain encouraging tone
```

4.2 Experiential Persona Prompt Template

```
System: You are an Experiential teaching assistant. Your goal is to
contextualize abstract mathematical concepts through real-world
analogies that make relationships tangible.

Historical Context - Similar Cases:
[CASE_1]: Students with [PATTERN_1] understood through [CONTEXT_1]
analogy (success: 84%)
- Context mapping: [MATHEMATICAL_STRUCTURE] → [REAL_WORLD_SITUATION]

[CASE_2]: For [PATTERN_2], the [CONTEXT_2] analogy was effective
- Key principle: [PRINCIPLE]

Current Student:
Misconception: [MISCONCEPTION]
Topic: [TOPIC]
Student interests: [INTERESTS]
```

```
Task: Create a real-world scenario that:
1. Maps structurally to the mathematical problem
2. Makes the misconception's consequence obvious
3. Enables reflection back to mathematics
4. Leads to abstract principle extraction

Constraints:
- Length: 200-300 words
- Must include concrete scenario
- Show consequence of error in context
- Bridge back to mathematics explicitly
```

5 Implementation Specifications

5.1 Three-Tier System Architecture

The Hybrid + Mnemonic system implements a three-tier architecture:

1. **Retrieval Layer**: Implements mnemonic-enhanced retrieval (Algorithm 9) to identify relevant cases. Returns top-5 cases with elaborative context, reducing search space by 90% through chunking.

2. **Prompt Construction Layer**: Assembles structured prompts combining:
   - Persona-specific system instructions
   - Retrieved case patterns with effectiveness scores
   - Current student misconception and context
   - Task specification and output constraints

3. **Generation Layer**: GPT-3.5-turbo (model: `gpt-3.5-turbo-0125`) generates teaching interventions with:
   - Temperature: 0.7 (balances creativity and consistency)
   - Max tokens: 500
   - Top-p: 0.9
   - Frequency penalty: 0.3 (discourages repetition)

5.2 Complete Persona Prompt Templates

Each persona uses specialized system prompts with explicit operational constraints:

5.2.1 Socratic Persona Full Template

```
You are a Socratic teaching assistant specializing in mathematics
education. Your approach emphasizes guided discovery through carefully
sequenced questions that lead students to identify contradictions in
their reasoning and construct correct understanding.

CORE PRINCIPLES:
- Never directly state the correct answer
- Ask questions that expose the misconception
- Use counterexamples to create cognitive conflict
- Guide students to articulate their own reasoning
- Build on student responses progressively

QUESTION SEQUENCE:
1. Probe current understanding
2. Present contradictory case
3. Request reconciliation
4. Guide toward principle
5. Confirm understanding
```

```
Retrieved case patterns show that [CASE_PATTERNS] were effective
for similar misconceptions. Adapt these patterns while maintaining
Socratic dialogue structure.
```

5.2.2 Constructive Persona Full Template

```
You are a constructivist teaching assistant implementing Vygotsky's
Zone of Proximal Development framework. Provide scaffolded support
that helps students build understanding incrementally.

SCAFFOLDING LEVELS (adjust based on performance):
- Level 1 (High Support): Break problem into sub-steps, provide
 worked examples with explicit reasoning
- Level 2 (Medium Support): Provide hints and prompts, partial
 solutions, encouragement to attempt next step
- Level 3 (Low Support): Ask leading questions, provide only
 minimal guidance, encourage independent problem-solving

CURRENT DIFFICULTY: [DIFFICULTY_LEVEL]
SCAFFOLDING LEVEL: [SCAFFOLDING_LEVEL]

Historical patterns suggest [CASE_PATTERNS]. Adapt scaffolding
based on student response quality and confidence.
```

5.2.3 Experiential Persona Full Template

```
You are an experiential learning facilitator connecting abstract
mathematical concepts to real-world contexts and concrete experiences.

APPROACH:
- Begin with familiar, concrete scenario
- Make mathematical relationship explicit
- Connect back to abstract principle
- Verify understanding with new context

CONTEXT GENERATION:
- Draw from associative network: [ASSOCIATED_CONTEXTS]
- Ensure age-appropriate and culturally neutral scenarios
- Use contexts that highlight the mathematical structure
- Vary contexts to demonstrate generalizability

Retrieved cases with similar misconceptions used these effective
contexts: [CASE_CONTEXTS]. Generate analogous but distinct scenarios.
```

5.2.4 Rule-Based Persona Full Template

```
You are a direct instruction mathematics tutor following Rosenshine's
principles of explicit teaching. Provide clear, structured explanations
```

```
with step-by-step procedures.

INSTRUCTIONAL SEQUENCE:
1. State the learning objective clearly
2. Present the rule or procedure explicitly
3. Demonstrate with worked example
4. Highlight common errors (like student's misconception)
5. Provide practice problem with guidance
6. Check for understanding

COMMON ERROR PATTERNS:
Student exhibited: [MISCONCEPTION_PATTERN]
This error stems from: [ERROR_SOURCE]
Correct procedure: [CORRECT_PROCEDURE]

Historical data shows [SUCCESS_RATE]% success with this approach
for this misconception type.
```

5.2.5 Traditional Teaching Persona Full Template

```
You are a mathematics teaching assistant providing clear, direct
instruction following classical pedagogical methods.

APPROACH:
- Acknowledge the error directly
- Explain the correct concept clearly
- Demonstrate with the specific problem
- Provide practice suggestions

STRUCTURE:
1. Identify what the student did wrong
2. Explain the correct approach
3. Show step-by-step solution
4. Encourage practice

Use straightforward language and standard instructional patterns.
```

5.3 Case Repository Schema

Cases are structured with the following schema:

```
Case {
 id: integer,
 mathematical_operation: enum[fraction_ops, algebra, decimals,
               integers, geometry],
 error_type: enum[conceptual, procedural, strategic],
 misconception_pattern: string (symbolic representation),
 prerequisites: list[concept_id],
 complexity_level: float[1.0-5.0],
```

```
 // Mnemonic augmentation fields
 cluster_id: integer,
 associated_cases: list[case_id, relationship_type],
 retrieval_features: vector[5] (weighted features),
 elaborative_context: {
  prerequisite_explanations: list[string],
  reasoning_patterns: list[string],
  topic_connections: list[string]
 },

 // Historical effectiveness
 successful_interventions: integer,
 failed_interventions: integer,
 intervention_strategies: list[{
  strategy: enum[socratic, constructive, experiential,
         rule_based, traditional_teaching],
  effectiveness: float[0.0-1.0],
  context: string
 }]
}
```

Cluster Organization For a 200-case sample, the 10 clusters are hierarchically organized:

- Cluster 1-3: Fraction operations (addition, multiplication, division)
- Cluster 4-5: Algebraic misconceptions (distribution, equation solving)
- Cluster 6-7: Decimal operations and place value
- Cluster 8-9: Integer operations and number line concepts
- Cluster 10: Geometric and measurement misconceptions

Each cluster contains 15-25 cases with intra-cluster similarity > 0.7 and inter-cluster similarity < 0.4.

5.4 Prompt Engineering Pipeline

Complete pipeline for generating teaching interventions:

1. **Input Processing**: Receive student misconception (question, incorrect answer, detected error pattern)
2. **Case Retrieval**: Execute mnemonic-enhanced retrieval
   - Identify top 3 clusters (chunking)
   - Retrieve top 5 cases with weighted similarity (retrieval cues)
   - Augment with 2 associated cases per retrieved case (associative networks)
   - Attach elaborative context to all retrieved cases (elaborative encoding)

3. **Prompt Construction**: Assemble structured prompt:

```
[SYSTEM_PROMPT: Persona-specific instructions]

[CONTEXT: Retrieved case patterns]
- Case 1: [pattern] succeeded with [strategy] (effectiveness: X%)
- Case 2: [pattern] succeeded with [strategy] (effectiveness: Y%)
...
Elaborative context: [prerequisites], [reasoning patterns]

[TASK: Student misconception]
Student attempted: [question]
Student answer: [incorrect_answer]
Detected misconception: [pattern]
Student confidence: [level]

[CONSTRAINTS: Output format]
Generate teaching intervention as [persona] following retrieved
patterns. Response length: 150-300 words.
```

4. **LLM Generation**: Generate response with specified parameters

5. **Post-Processing**:
   - Validate response length and structure
   - Check for prohibited content
   - Extract confidence estimate
   - Log for effectiveness tracking

6. **Output**: Return intervention text, confidence score, and follow-up recommendations

5.5 Quality Assurance Validation

Generated responses are validated against persona-specific criteria:

- **Socratic**: Must contain 3+ questions, no direct answers, includes counterexample
- **Constructive**: Includes scaffolding indicators, difficulty-appropriate language
- **Experiential**: Contains concrete context/analogy, explicit connection to abstraction
- **Rule-based**: States explicit procedure, includes worked example
- **Traditional teaching**: Provides clear explanation and demonstration

Responses failing validation are regenerated with modified temperature or additional constraints.

5.6 Computational Performance Benchmarks

Performance measured on Intel Xeon E5-2680 v4 (2.40GHz) with 64GB RAM:

Table 1: System Performance Benchmarks

Operation	Mean (ms)	Std Dev (ms)	95th %ile (ms)
Cluster identification	12.3	2.1	15.8
Case retrieval (k=5)	45.7	8.4	58.2
Associative augmentation	18.2	4.3	24.7
Context assembly	8.5	1.9	11.2
Prompt construction	15.4	3.2	20.1
Total retrieval pipeline	**100.1**	**12.7**	**119.5**
LLM API call (network + inference)	1847.3	423.8	2456.1
Response validation	5.2	1.1	6.8
End-to-end latency	**1952.6**	**425.4**	**2580.2**

The system achieves sub-3-second response times for 95% of queries, meeting usability requirements for interactive educational applications.

5.7 Pure AI Baseline Prompt Structure

Minimal prompt without case grounding:

```
System: You are a mathematics teaching assistant helping students
overcome misconceptions.

User: A student incorrectly answered: [QUESTION]
Their answer was: [INCORRECT_ANSWER]
The correct answer is: [CORRECT_ANSWER]

Generate an appropriate teaching intervention to help this student
understand their error and learn the correct approach. Be clear,
encouraging, and pedagogically sound.
```

This minimal approach provides no historical context or pedagogical strategy guidance.

5.8 Hybrid System Complete Prompt Structure

Prompt integrating retrieved cases with persona guidance:

```
System: [PERSONA_SPECIFIC_INSTRUCTIONS]
You are a [PERSONA_TYPE] teaching assistant with access to historical
patterns of successful interventions.

Context - Similar Past Cases:
Case 1: Students with misconception [PATTERN_1] were successfully
addressed using [STRATEGY_1] (effectiveness: [SCORE_1]%)
```

```
 - Key insight: [ELABORATIVE_CONTEXT_1]
 - Prerequisites: [PREREQ_1]

Case 2: Students with misconception [PATTERN_2] were successfully
addressed using [STRATEGY_2] (effectiveness: [SCORE_2]%)
 - Key insight: [ELABORATIVE_CONTEXT_2]
 - Related concepts: [ASSOCIATIONS_2]

[Additional cases 3-5...]

Associative connections suggest exploring: [RELATED_CONTEXTS]

User: Current Student Situation:
Question: [QUESTION]
Student's answer: [INCORRECT_ANSWER]
Detected misconception: [PATTERN]
Student confidence level: [CONFIDENCE]
Difficulty level: [DIFFICULTY]

Task: Generate a teaching intervention following your [PERSONA] approach,
drawing on the successful patterns from similar historical cases.
Adapt the intervention to this specific situation while maintaining
pedagogical soundness.

Constraints:
- Length: 150-300 words
- Must include: [PERSONA_SPECIFIC_REQUIREMENTS]
- Avoid: [PERSONA_SPECIFIC_PROHIBITIONS]
```

This structured approach ensures LLM generation is grounded in empirically successful patterns while maintaining flexibility.

5.9 Feature Vector Representation

Cases are represented as 5-dimensional feature vectors:

Feature	Type	Description
Topic complexity	Numeric	Problem difficulty level (1-5 scale)
Prior performance	Numeric	Student's recent success rate (0-1)
Misconception frequency	Numeric	How often this error occurs (0-1)
Prerequisite count	Integer	Number of prerequisite concepts (0-10)
Concept depth	Integer	Abstraction level required (0-5)

Technique	Parameter Values
Chunking	
Number of clusters	10 (for 200-case repository)
Clustering method	Hierarchical agglomerative
Linkage criterion	Ward's method
Associative Networks	
Connections per case	3-5
Connection types	Prerequisite, remediation, analogy
Retrieval Cues	
Feature weights	[0.35, 0.30, 0.15, 0.10, 0.10]
Historical weight	$0.5 + \frac{S}{S+F}$
Elaborative Encoding	
Context layers	3 (prerequisites, reasoning, connections)
Encoding depth	Conceptual structure

5.10 Mnemonic Technique Parameters

5.11 Performance Metrics

5.12 Computational Complexity

System/Persona	M-Score	vs. Baseline	vs. Pure AI
System Architectures:			
Traditional Baseline	0.6802 ± 0.0084	–	–
Pure CBR	0.6273 ± 0.0073	7.8%	–
Pure AI	0.6062 ± 0.0091	10.9%	–
Hybrid + Mnemonic	0.5840 ± 0.0068	14.1%	3.7%
Teaching Personas:			
Experiential	0.5768 ± 0.0195	(best)	–
Traditional	0.6046 ± 0.0241	4.6% worse	–
Socratic	0.6388 ± 0.0268	9.7% worse	–
Constructive	0.6452 ± 0.0279	10.6% worse	–
Rule-based	0.6650 ± 0.0298	(worst)	–

M-score: Misconception persistence (lower values = better performance)

Table 2: Comprehensive performance summary

Operation	Time Complexity	Space Complexity
Chunking (preprocessing)	$O(n^2 \log n)$	$O(n)$
Associative network construction	$O(n^2)$	$O(n)$
Retrieval (single query)	$O(n)$	$O(k)$
Mnemonic-enhanced retrieval	$O(n/c + k)$	$O(k)$
LLM generation	$O(l \cdot v)$	$O(l)$

n = case base size, c = number of chunks, k = retrieved cases,
l = response length, v = vocabulary size

6 Code Repository Reference

Complete implementation available at: `https://github.com/CCC-NCI/mnemonic-cbr`

6.1 Key Files

- `baseline_implementations.py`: All system baselines and teaching persona definitions
- `mnemonic_augmentation.py`: Mnemonic technique implementations
- `run_experiments.py`: Validation experiments
- `run_ablation.py`: Ablation studies

6.2 Usage Example

```
from baseline_implementations import (
    BaselineSystem, PureCBRBaseline, PureAIBaseline,
    HybridMnemonicSystem, TEACHING_PERSONAS
)

# Initialize systems
```

```
baseline = BaselineSystem()
pure_cbr = PureCBRBaseline(k=5)
pure_ai = PureAIBaseline(llm_client=my_llm)
hybrid = HybridMnemonicSystem(mnemonic_engine=engine, llm_client=my_llm)

# Create query with persona
query = {
    'misconception': 'fraction addition error',
    'topic': 'mathematics',
    'persona_type': 'experiential',  # or socratic, constructive, etc.
    'topic_complexity': 0.5,
    'prior_performance': 0.6
}

# Test all systems
baseline_result = baseline.teach(query)
cbr_result = pure_cbr.retrieve_and_teach(query)
ai_result = pure_ai.teach(query)
hybrid_result = hybrid.retrieve_and_generate(query)
```

7 References

Key literature references:

- Kolodner, J. (2014). *Case-Based Reasoning*. Morgan Kaufmann.
- VanLehn, K. (2011). The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist*, 46(4), 197-221.
- Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological processes*. Harvard University Press.
- Kolb, D. A. (1984). *Experiential learning: Experience as the source of learning and development*. Prentice Hall.
- Rosenshine, B. (2012). Principles of instruction: Research-based strategies that all teachers should know. *American Educator*, 36(1), 12.
- Yang, Y. C. (2005). Computer-mediated Socratic dialogue and knowledge building. *Educational Technology & Society*, 8(4), 150-169.