A Comparative Analysis of Prompt Engineering in Large Language Models

Presenter: Deepti Dabral

Presented on: April 11, 2025

Key References:

A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications

Pranab Sahoo[1] , **Ayush Kumar Singh**[1] , **Sriparna Saha**[1] , **Vinija Jain**[2,3] , **Samrat Mondal**[1] and **Aman Chadha**[2,3]

[1]Department of Computer Science And Engineering, Indian Institute of Technology Patna

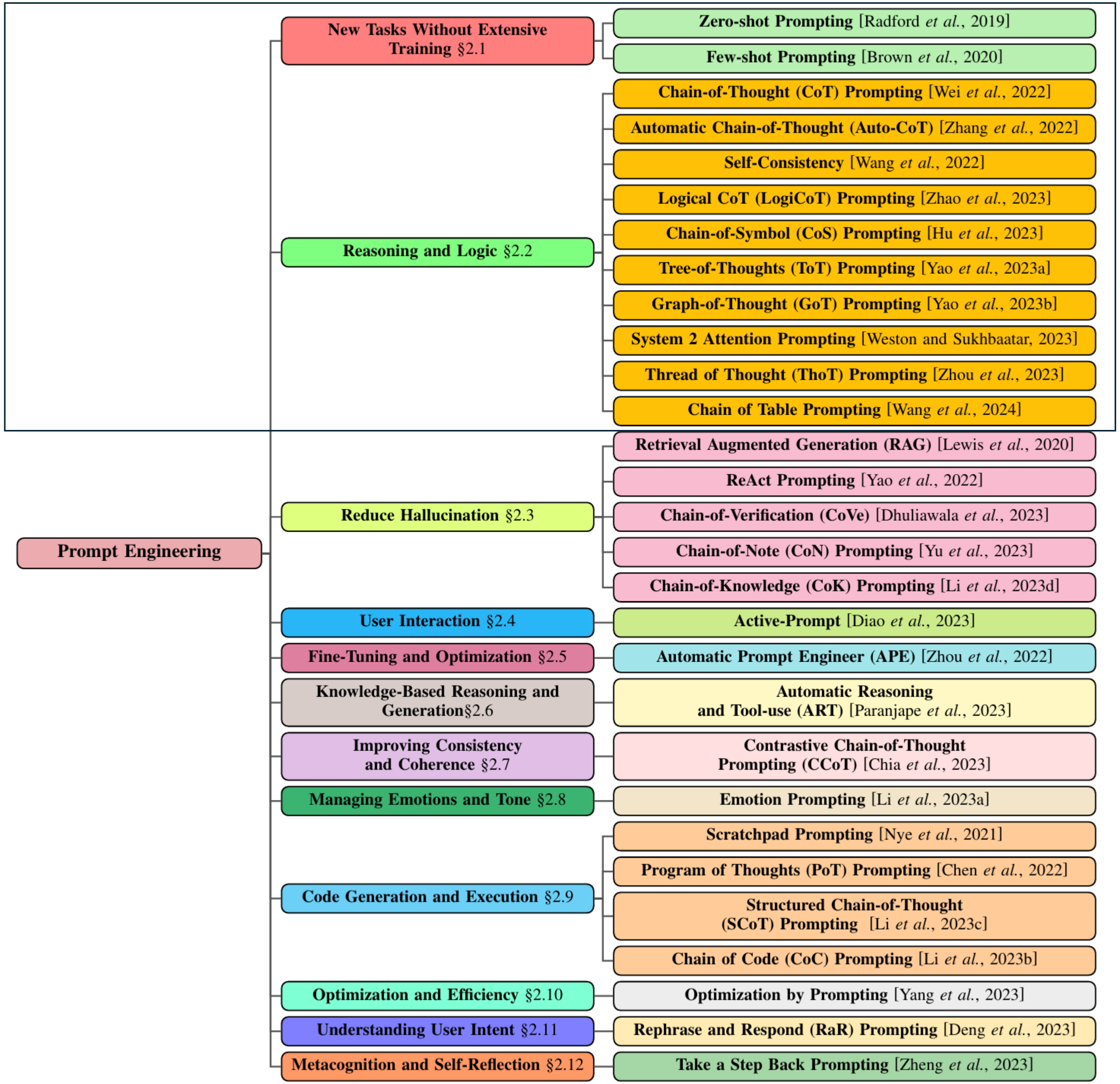
[2]Stanford University, [3]Amazon AI

{pranab_2021cs25, ayush_2211ai27, sriparna, samrat}@iitp.ac.in, hi@vinija.ai, hi@aman.ai

Table of Contents

Focus for today

Taxonomy of Prompt Engineering Techniques in LLMs



Source: A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications

Table of Contents

1: New Tasks Without Extensive Training

Use Case	1.1 Zero-Shot Prompting	1.2 Few-Shot Prompting
1. Fraud Scenario Simulation / Adversarial testing for a payment provider	Generate a scenario of a credit card fraud attempt that could bypass the organization’s fraud detection models.	Here are examples of sophisticated fraud patterns : • Eg1: A fraudster made small purchases at gas stations to test card validity and then made a series of mid-size electronics purchases at multiple retailers within ~2-3 hours. • Eg2: Multiple bank cards were used from the same device ID to make similar purchases on travel websites, rising in value over 3 days, with all purchases occurring at night. Following these patterns, generate new fraud scenarios to evade detection.
2. Feedback Integration in credit risk monitoring systems	Summarize the key patterns in this feedback provided by the credit risk expert and identify which model components should be adjusted based on it.	Here are examples of how credit analyst feedback is incorporated into our risk models: • Eg1: Feedback: "Small business customers with seasonal income are being flagged as high risk during their off-season months despite perfect payment history.” Model Adjustment: Added seasonality factors to cash flow analysis and reduced the weight of current month income for businesses with historical seasonal patterns. • Eg2: Feedback: "Customers with recent address changes are receiving higher risk scores despite other positive indicators.” Model Adjustment: Implemented a 90-day grace period where address changes have reduced impact if other indicators are stable. Similarly, analyze the new expert feedback and recommend the model adjustments.
3. Driving explainability in product recommender systems in retail banking	Explain to a customer why our system has recommended the Premium Travel Card over the Cash Back Card based on their spending patterns.	Here are examples of product recommendation explanations: • Eg1: Customer query: "Why are you recommending this portfolio?“ Response: “Because it provides moderate growth potential and is aligned with your risk tolerance." • Eg2: Customer query: "Why the travel insurance add-on?“ Response: "Based on your upcoming travel, the travel insurance add-on would provide coverage for all three trips at $240 total, which is much lower than purchasing separate coverage for each trip." Similarly, explain why we're recommending the Family Protection Insurance to a customer.

2: Reasoning and Logic

Use Case	2.1 Chain-of-Thought (CoT) Prompting	2.2 Automatic Chain-of-Thought (Auto-CoT) Prompting
1. Fraud Scenario Simulation / Adversarial testing for a payment provider	Think step by step how to develop a fraud scenario to bypass fraud detection models. 1. Consider the typical patterns that fraud detection systems look for. 2. Think about how those patterns could be deliberately avoided. 3. Consider data points (variables / features) that model can access. 4. Develop a scenario that appears legitimate across those variables. 5. Detail how the fraudster would execute this. 6. Explain why it might be difficult to detect.	Q1: How would you design a fraudulent transaction to evade modern payment fraud detection systems? Let's think step-by-step: 1. First, understand what triggers modern fraud detection systems. 2. Then, analyze how these triggers could potentially be avoided. 3. Finally, design a pattern that minimizes detection likelihood. Q2: Let's think step-by-step: Q3: Let's think step-by-step: Now, using similar reasoning to answer the following: How would you design fraudulent transactions to evade fraud detection at a specific organization? Let's think step-by-step:
2. Feedback Integration in credit risk monitoring systems	Let's think through how to effectively integrate customer feedback into our credit risk model: 1. Analyze the feedback data to identify key themes of model deterioration or failure. 2. Categorize the feedback based on which variables they relate to (e.g., income, payment history, debt). 3. Evaluate whether the feedback suggests a systematic bias or error in the model. 4. Determine the impact of modification. 5. Prioritize changes based on potential impact and implementation complexity. 6. Suggest parameter adjustments or feature edits. 7. Conduct validation tests to ensure the changes improve outcomes.	Q1: How would a bank identify weaknesses in their mortgage default prediction model? To answer this, let's break it down into steps: 1. First, understand what feedback is telling us about the model. 2. Then, identify which model components are affected. 3. Next, evaluate potential adjustments to these components. 4. Finally, consider how to implement and validate these changes. Q2: Let's think step-by-step: Q3: Let's think step-by-step: Now, using similar reasoning to answer the following: How should a credit risk monitoring system integrate feedback from credit analysts? Let's think step-by-step:

2: Reasoning and Logic

Use Case	2.1 Chain-of-Thought (CoT) Prompting	2.2 Automatic Chain-of-Thought (Auto-CoT) Prompting
3. Driving explainability in product recommender systems in retail banking	Explain to a customer why we're recommending a specific product (e.g., 529 plans, UTMA accounts) in a stepwise manner as illustrated below. 1. Acknowledge the customer's recent life changes (e.g., becoming a parent) known to us. 2. Explain how these changes affect their financial needs. 3. Outline the specific components of the product offered. 4. Connect each component to customer’s needs. 5. Compare this recommendation to alternatives they might be considering. 6. Explain the cost-benefit analysis in simple terms. 7. Summarize why this is the best product for their financial objectives.	How would you explain a complex financial product recommendation to a customer? Let’s think step-by-step: 1. First, understand the customer's needs and context. 2. Then, identify the key product features to address these needs. 3. Next, translate technical benefits into practical advantages. 4. Finally, compare with alternatives to justify the recommendation. Q2: Let’s think step-by-step: Q3: Let’s think step-by-step: Now, using similar reasoning to answer the following: How should a customer service chatbot generate clear narrative explanations for why a specific banking product is recommended to a customer?

2: Reasoning and Logic

Use Case	**2.3 Self-Consistency**	**2.4 Logical Chain of Thought Prompting**
1. Fraud Scenario Simulation / Adversarial testing for a payment provider	Generate n different fraud scenarios that could bypass a fraud detection system. For each of them: 1. Detail the transaction patterns 2. Explain the technical approach used to avoid detection 3. Identify potential vulnerabilities in the scenario 4. Rate its likelihood of success Finally, analyze all to identify which one appears most likely to succeed and why. Then flesh it out further for execution.	Premises: 1. Modern fraud detection systems flag deviations from established customer behavior. 2. These systems monitor transaction frequency, amount, merchant category, and location. 3. Risk scores are based on the combination of these factors. 4. Thresholds trigger manual review or auto-decline of transactions. The logic derived from these premises is: 1. A fraudster must mimic normal behaviour to avoid detection. 2. Normal-looking patterns reduce alerts for individual anomalies. 3. Gradual increases in transaction amounts help avoid suspicion. 4. Using familiar merchants avoids category-based flags. Therefore, develop a fraud scenario where a fraudster: 1. Start with mimicking normal transaction patterns. 2. Gradually adjust these patterns over time. 3. Build a justifiable explanation that explain the pattern shifts. 4. Execute large, ambitious frauds after building trust with the system.

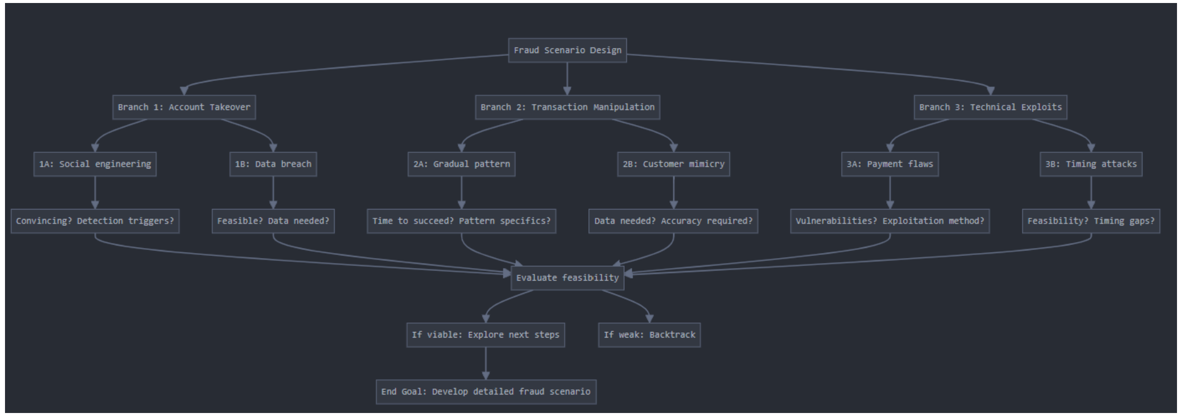
2: Reasoning and Logic

Use Case	2.3 Self-Consistency	2.4 Logical Chain of Thought Prompting
2. Feedback Integration in credit risk monitoring systems	Analyze the analyst feedback dataset in 5 different ways. For each way: 1. Group feedback by model component and identify most common issues. 2. Evaluate feedback based on customer segments affected and quantify the business impact. 3. Categorize feedback based on whether it suggests a FP / FN problem. 4. Map feedback to specific model features and scoring thresholds. 5. Examine temporal patterns in the feedback to identify any trends. Finally, synthesize the findings to identify consistent patterns across different analytical approaches. Then develop a prioritized list of model adjustments that address the most critical and consistent issues.	Premises: 1. Credit risk models assign weights to factors like payment history, utilization, income, and debt-to-income ratio. 2. Analysis indicates that certain customer segments get risk scores that don’t match their true repayment likelihood. 3. Model adjustments must fix bias without losing predictive power. 4. Changes must be validated against historical data. The logic derived from these premises is as follows: 1. Consistent feedback on a specific customer segment suggests a bias. 2. If the bias is tied to a specific feature, then adjust its weight. If multiple features are involved, then a comprehensive modification is needed. 3. Significant impact on model performance requires a detailed inspection. Therefore, analyze the attached feedback dataset to: 1. Identify the patterns in model overrides by human (analyst) feedback. 2. Link each override pattern to the premise. 3. Develop logical modifications to model parameters. 4. Develop a validation framework using historical outcomes.

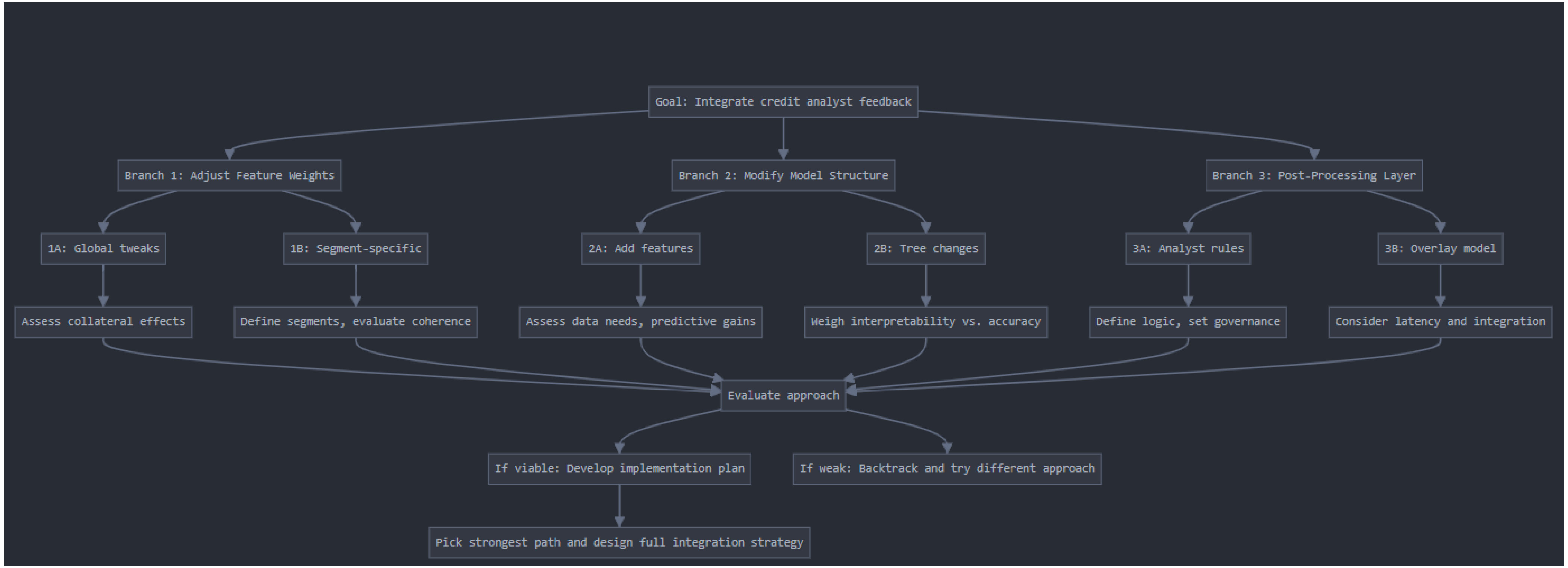
2: Reasoning and Logic

Use Case	2.3 Self-Consistency	2.4 Logical Chain of Thought Prompting
3. Driving explainability in product recommender systems in retail banking	Generate 3 different rationales for recommending the Premium Health Package to a family of four with two young children. For each explanation: 1. Focus on comprehensive coverage and peace of mind. 2. Emphasize cost-effectiveness over potential out-of-pocket expenses. 3. Highlight flexibility and customization options as children grow. For each of them, include: • A friendly opening • Top benefits specifically relevant to target customer profile • Summary of cost considerations • A comparison with alternatives Finally, evaluate which one provides the clearest justification. Refine it further for final use.	Premises: 1. The recommended product plan offers market downturn protection, guaranteed minimum returns, and flexible withdrawals. 2. The customer worries about market volatility and uncertain retirement. 3. The customer has moderate risk tolerance and prefers capital preservation. 4. Alternative products offer either higher returns with less protection or similar protection at higher costs. The logic derived from these premises is as follows: 1. Volatility concerns make downside protection essential. 2. Uncertain timelines call for flexible withdrawals. 3. Moderate risk tolerance needs both protection and some growth. 4. Given less favorable trade-offs, the recommended plan is best. Thus, explain to the customer: 1. How each product feature addresses their concerns? 2. Why this match is superior to alternatives? 3. How is the cost justified by the specific benefits? 4. The timeline for when they should reconsider this.

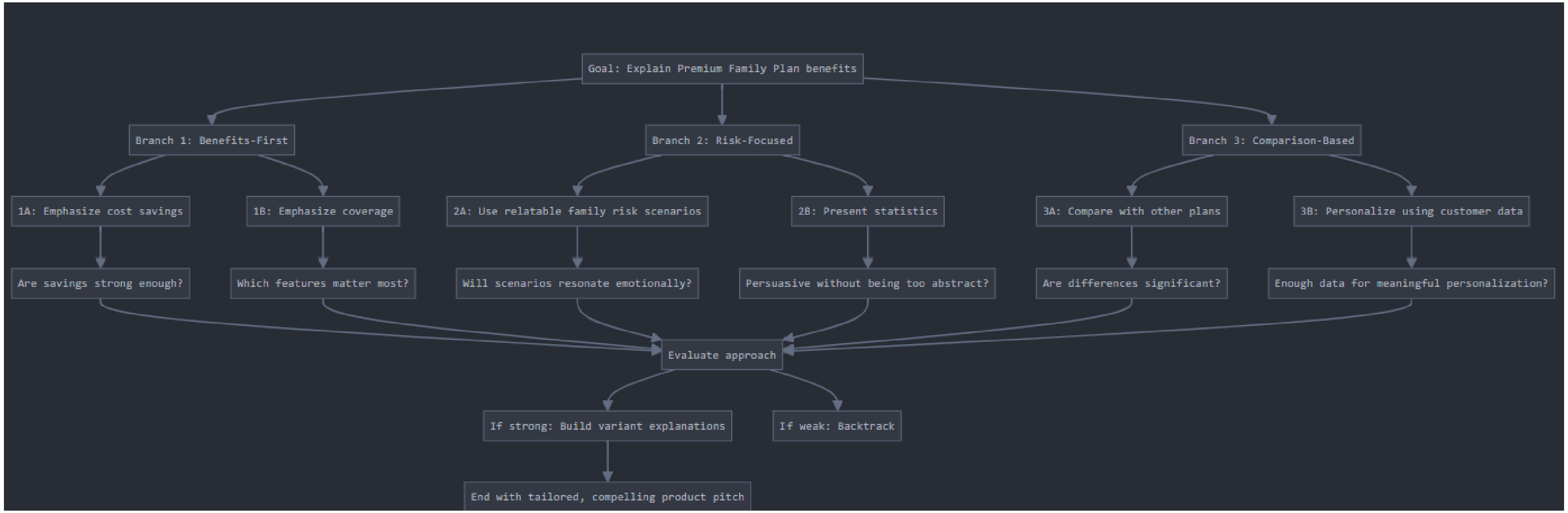
2: Reasoning and Logic

Use Case	2.5 Chain-of-Symbol (CoS) Prompting	2.6 Tree-of-Thoughts (ToT) Prompting
1. Fraud Scenario Simulation / Adversarial testing for a payment provider	Symbolic framework: - C = legitimate customer behavior - T = transaction (amount, merchant, location, time) - A = anomaly detection threshold - D = probability of fraud - FP = fraud pattern Define baseline: $C = \{T_1, T_2, ..., T_n\}$ where each T_i is within normal parameters To minimize detection: $D(FP) \propto \sum\|FP - C\|$ Steps: $FP_0 = C \longrightarrow FP_t = FP_{t-} + \delta_t \longrightarrow \underset{\|\delta_t\| < A}{\overset{D(FP) \propto}{\sum}} \longrightarrow T$ Match Legit Behavior — Gradual Deviation — Optimize sequence — high-value electronics purchase Scenario: Start: FP_0 = small-size retail transactions. Goal: High-value electronics purchase. Constraint: Gradual deviation is undetectable. Task: Show symbolic steps. Translate to real-world.	

2: Reasoning and Logic

Use Case	2.6 Tree-of-Thoughts (ToT) Prompting
2. Feedback Integration in credit risk monitoring systems	

2: Reasoning and Logic

Use Case	2.6 Tree-of-Thoughts (ToT) Prompting
3. Driving explainability in product recommender systems in retail banking	

2: Reasoning and Logic

Use Case	2.7 Graph-of-Thought (GoT) Prompting	2.8 System2Attention Prompting
1. Fraud Scenario Simulation / Adversarial testing for a payment provider	Develop a fraud scenario using a graph where interconnected elements influence each other. Sample nodes could be Transaction Patterns (TP), Geographic Locations (GL), Merchant Categories (MC), Time Patterns (TiP), Amount Progression (AP), and more. Edges (influence): • TP → GL: Pattern-location correlation • TP → MC: Pattern-merchant linkage • TiP → AP: Timing drives amount • MC → AP: Merchant limits amount Instructions: • Assign initial states to key nodes • Propagate constraints via edges • Identify valid, coherent fraud paths • Expand the strongest path into a detailed scenario Using this methodology, develop a fraud scenario that reflects complex interdependencies, intended to bypass detection models, through a coordinated evolution of TP, GL, MC, TiP, and AP.	Design a fraud scenario using the following approach: System 1 (Fast Thinking): Generate overview of 5 distinct fraud approaches to bypass detection systems. System 2 (Analytical Thinking): For each approach, identify likely detection triggers, assign risk scores, evaluate feasibility and required resources, and highlight high-risk elements needing deeper focus. Attention Focus Areas: • Transaction sequencing • Geographic consistency • Amount progression • Timing patterns • Authentication bypass Integrate the strongest elements into a fraud scenario to: 1. Minimize detection probability 2. Address attention areas 3. Include detailed transaction flow (sequence, timing, amount, channel, merchants) 4. Exploit known system vulnerabilities

2: Reasoning and Logic

Use Case	2.7 Graph-of-Thought (GoT) Prompting	2.8 System2Attention Prompting
2. Feedback Integration in credit risk monitoring systems	Build a feedback integration system as a dependency graph. Sample nodes would be Data Features (DF), Model Components (MC), Risk Thresholds (RT), Analyst Feedback (AF), Customer Segments (CS), Performance Metrics (PM), Complexity (C) Edges (influences): AF → MC: Feedback modifies components CS → AF: Segments influence feedback types MC → PM: Component changes affect performance DF → MC: Features shape components MC → RT: Components alter thresholds Instructions: 1. Node States: Define valid states per node (e.g., MC: tuned, outdated, retrained). 2. Edge Propagation: Specify directional effects (e.g., AF ↑ → MC adjusts). 3. Mapping: Link source node (AF) directly to target / impacted node (MC). 4. Impact Tracing: Trace effects across the graph. 5. Optimization: Identify changes that maximize performance (PM) and minimize complexity ©. 6. Plan: Summarize model updates based on graph. Using this methodology, develop a feedback-driven model adjustment plan optimized for performance and interpretability.	Design a feedback integration strategy using the following approach: System 1 (Fast, Intuitive): Identify ~10 potential patterns in the analyst feedback data. System 2 (Slow, Analytical): For each pattern: • Measure feedback frequency and consistency. • Assess potential impact on model performance. • Estimate implementation complexity and resources. • Calculate expected improvement in prediction accuracy. Attention Focus Areas: • Statistical significance • Segment performance • Feature importance • Validation methodology Develop a feedback integration plan that: 1. Prioritizes changes based on impact. 2. Defines specific model adjustments. 3. Develops a clear implementation roadmap. 4. Establishes rigorous validation protocols. Present the complete integration strategy.

2: Reasoning and Logic

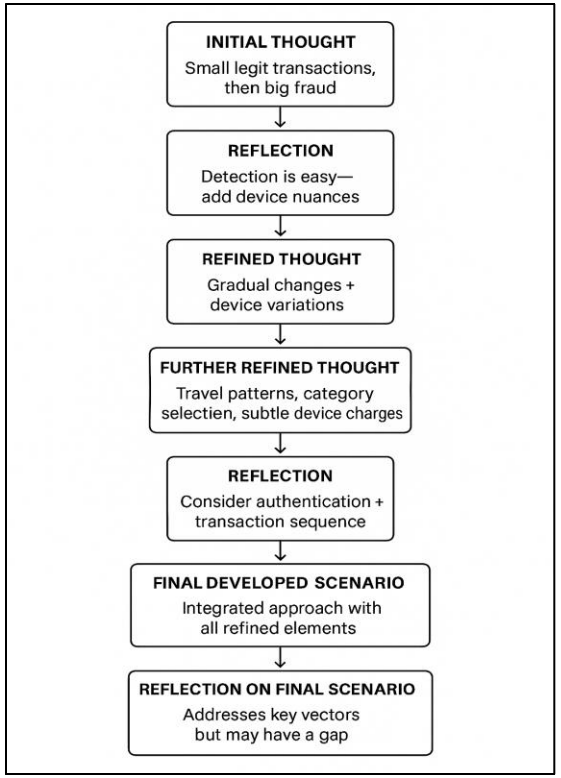
Use Case	2.7 Graph-of-Thought (GoT) Prompting	2.8 System2Attention Prompting
3. Driving explainability in product recommender systems in retail banking	Create a product explanation using a graph. Sample nodes would be Product Features (PF), Customer Needs (CN), Financial Considerations (FC), Explanation Components (EC), Decision Factors (DF), etc. Edges (influences): PF → CN: How product features fulfill customer needs; PF → FC: How product features justify financial considerations, and others. Instructions: 1. For each node, develop multiple states based on the customer profile. 2. For each edge, define how the source node's state should influence the explanation strategy. Using this methodology, develop an optimal explanation by: 1. Identifying customer needs 2. Mapping product features to needs 3. Determining the most compelling connections and explanation paths 4. Crafting a coherent narrative that follows these paths	Design a product explanation logic using the following approach: System 1 (Fast Thinking): Rapidly generate 5 explanations highlighting core features, and product benefits. System 2 (Analytical Thinking): For each angle: • Assess fit with customer’s needs • Quantify financial advantages • Compare with alternative plans • Identify resonance and potential objections Attention Focus Areas: • Personalized cost-benefit analysis • Coverage match to family concerns • Long-term value demonstration • Clear comparative advantages • Anticipation of objections Craft a detailed explanation that: 1. Starts with a strong, engaging hook 2. Embeds tailored analytical insights 3. Highlights financial value for the family 4. Addresses potential concerns

2: Reasoning and Logic

Use Case	2.9 Thread of Thought (ThoT) Prompting	2.10 Chain of Table Prompting

Use Case: 1. Fraud Scenario Simulation / Adversarial testing for a payment provider

2.9 Thread of Thought (ThoT) Prompting



This is a single path example but there can be multiple threads each indicating a different fraud scenario. For multi-thread reasoning paths, each thread is evaluated for quality and coherence. And the best elements across are consolidated into a final solution.

2.10 Chain of Table Prompting

Design a fraud scenario using tabular reasoning. Tables being:

1. Legitimate Customer Profile - Capture typical transaction patterns (amount, frequency, category, geography, time).
2. Detection Matrix - List alert thresholds and weights for key risk signals (amount spike, geo shift, velocity, etc.).
3. Fraud Progression Strategy - Outline evolving fraud tactics across 4 stages (types, amount, geo, timing, authentication).
4. Risk Assessment by Stage - Estimate detection risk, highlight triggers, and suggest evasion tactics per stage.

Instructions:

1. Populate all tables with realistic values.
2. Ensure logical escalation across stages.
3. Minimize detection by aligning with baseline and detection matrix.
4. Conclude with a narrative walkthrough of the fraud plan.

Table 1: Baseline Legitimate Customer Behavior

Transaction Type	Typical Amount Range	Frequency	Merchant Categories	Geographic Pattern
Grocery	$50-100	Weekly	Supermarkets	Near home
Dining	$30-100	2-3x/week	Restaurants	Within 20 miles
Retail	2ßxmonth	2x/month	Department stores	Weekends
Travel	$500-2000	Quarterly	Airlines, hotels	Seasonal
Subscription	$10-50	Monthly	Digital services	1st of month

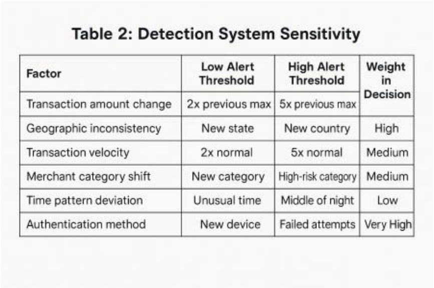


Table 2: Detection System Sensitivity

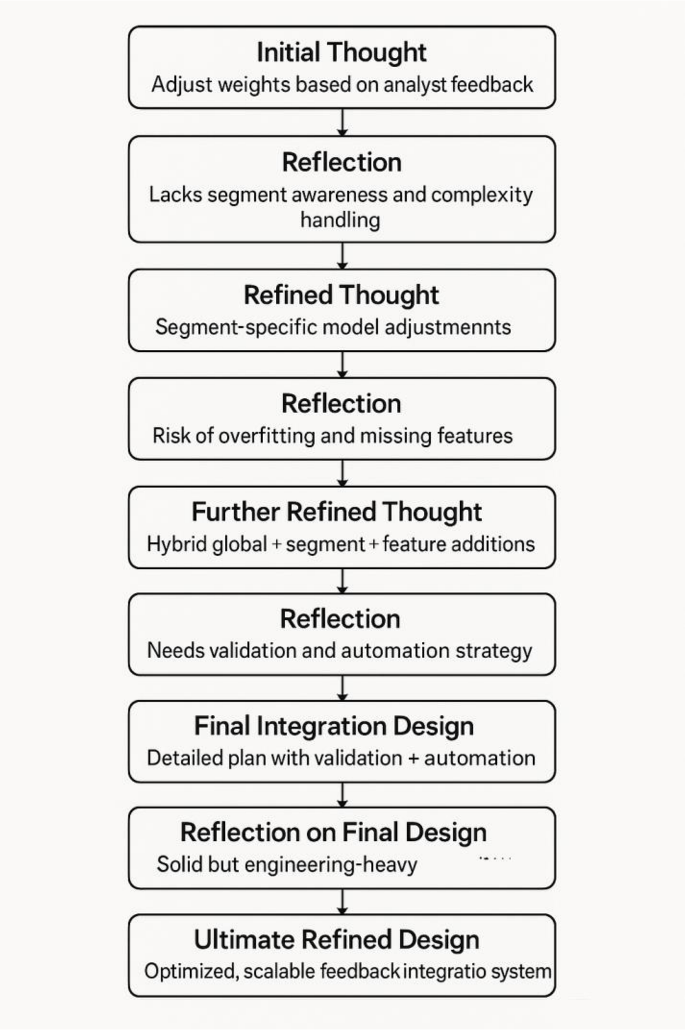
Factor	Low Alert Threshold	High Alert Threshold	Weight in Decision
Transaction amount change	2x previous max	5x previous max	
Geographic inconsistency	New state	New country	High
Transaction velocity	2x normal	5x normal	Medium
Merchant category shift	New category	High-risk category	Medium
Time pattern deviation	Unusual time	Middle of night	Low
Authentication method	New device	Failed attempts	Very High

2: Reasoning and Logic

Use Case

2. Feedback Integration in credit risk monitoring systems

2.9 Thread of Thought (ThoT) Prompting



2.10 Chain of Table Prompting

Build a feedback-driven model improvement plan using 4 structured tables.

1. Feedback Summary: Capture top feedback categories, their frequency, affected segments, and model components.
2. Component Impact: Assess each component's weight, feedback direction, statistical significance, and adjustment.
3. Segment Impact: Track false positive/negative rates, expected improvement, and priority for each segment.
4. Implementation Plan: Outline phased changes, complexity, validation, success metrics, and timelines.

Instructions:

1. Populate all tables with values.
2. Prioritize actions by impact vs. effort.
3. Define phased rollout with code-level changes.

Table 1
Feedback Categories and Frequencies

Feedback Category	Frequency	Customer Segments Affected	Model Components Implicated
[Category 1]	[Count]	[Segments]	[Components]
[Category 2]	[Count]	[Segments]	[Components]
[Category 3]	[Count]	[Segments]	[Components]
[Category 4]	[Count]	[Segments]	[Components]
[Category 5]	[Count]	[Segments]	[Components]

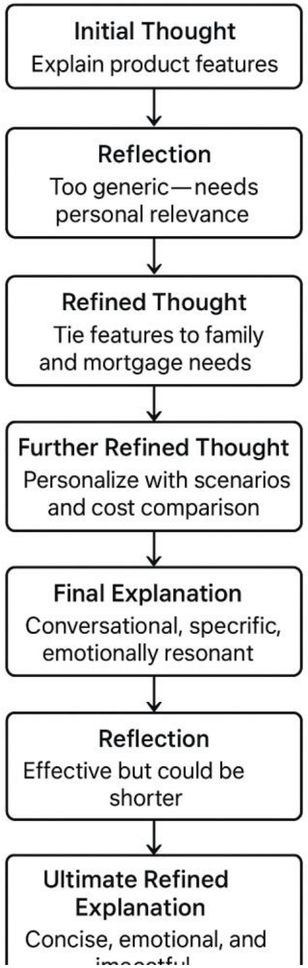
Table 2: Model Component Analysis

Model Component	Current Weight	Feedback Impact Direction	Statistical Significance	Adjustment Approach
[Component 1]	[Weight]	Increase/ Decreasee	[p-value]	[Approach]
[Component 2]	[Weight]	Increase/ Decreasee	[p-value]	[Approach]
[Component 3]	[Weight]	Increase/ Decreasee	[p-value]	[Approach]
[Component 4]	[Weight]	Increause/ Decreasee	[p-value]	[Approach]
[Component 5]	[Weight]	Increause/ Approauch	[Approah]	[Approach]

2: Reasoning and Logic

Use Case | **2.9 Thread of Thought (ThoT) Prompting** | **2.10 Chain of Table Prompting**

3. Driving explainability in product recommender systems in retail banking



Build a personalized, data-driven, emotionally resonant product explanation using structured tables.

1. Customer Profile: Link attributes to benefits and prioritize messaging.
2. Feature Mapping: Personalize features and show competitive edge.
3. Cost Analysis: Compare financial impact with vs. without the plan.
4. Explanation Flow: Align key messages with data and emotion.

Instructions:

1. Fill each table with personalized, data-driven content based on customer inputs.
2. Develop a script that walks through these tables.
3. Ensure emotional and rational balance — align facts with empathy.
4. Iterate benefits to tackle objections.

Table of Contents

1: New Tasks Without Extensive Training

Dimension	1.1 Zero-Shot Prompting	1.2 Few-Shot Prompting
What is it?	• Direct task specification without exemplars • Single-pass instruction interpretation • Relies entirely on pre-trained knowledge encoding	• Exemplar-based inductive learning enabling pattern recognition • Implicit meta-learning through context
How it works?	Linear unidirectional processing: Instruction → LLM → Output	Augmented linear processing: Instruction + Examples_1...k + Query → LLM → Output
Tasks well suited for	• Well-defined tasks with clear instructions • Tasks with significant representation in training data • Limited effectiveness for complex reasoning	• Tasks benefiting from concrete examples • Distribution shift scenarios
Implementation Considerations	• Heavily dependent on instruction wording and specificity • Requires precise task specification • High sensitivity to instruction phrasing	• Context window limitations restrict example count • Example selection critically impacts performance • Requires careful ordering and formatting of examples
Observations about performance	• High variance across different task formulations • Degrades rapidly with task complexity • Vulnerable to misinterpretation of ambiguous instructions	• Performance typically increases with number of examples until saturation • Strong ordering effects observed (recency bias) • Demonstrates in-context learning abilities
Prompt Acquisition	Manual	Manual
Prompt Turn	Single	Single

2: Reasoning and Logic

Dimension	2.1 Chain-of-Thought (CoT) Prompting	2.2 Automatic Chain-of-Thought (Auto-CoT)
What is it?	• Explicit reasoning decomposition • Step-by-step intermediate computation • Verbalized problem-solving process	• Self-generated decomposition • Two-phase generation: question clustering and reasoning • Automated exemplar creation
How it works?	Sequential flow: Instruction → [Reasoning Path]: Step#1, Step#2, …, Step#n → Output	Bootstrapped Q&A: Question → [Exemplar (eg) Generation] → [Reasoning Path] → Output
Tasks well suited for	• Multi-step reasoning tasks • Mathematical problem-solving • Logical deduction and inference	• Diverse question answering • Domains lacking human-annotated reasoning steps • Scalable reasoning across multiple tasks
Implementation Considerations	• Requires models with sufficient reasoning capacity • Sensitive to reasoning path formulation • Context window must accommodate entire reasoning chain	• Requires effective question clustering algorithms • Two-stage process increases complexity • Potential error propagation from one step to another
Observations about performance	• Improves performance on complex reasoning tasks and can solve intractable problems • Provides interpretability and error diagnosis	• Approaches manual CoT performance w/o human annotation • More robust across diverse question types • Reduces prompt engineering effort but higher variance
Prompt Acquisition	Manual	LM generated
Prompt Turn	Multi	Multi

2: Reasoning and Logic

Dimension	2.3 Self-Consistency	2.4 Logical Chain of Thought Prompting
What is it?	• Stochastic sampling with aggregation • Multiple reasoning paths with majority voting • Ensemble-based error correction	• Formal logic-based reasoning • Structured logical operators and rules • Think-verify-revise loop
How it works?	• Parallel diversified paths: Instruction → [Path1, Path2, ..., Pathn] → Aggregation → Output	• Directed acyclic graph of propositions: Premises → [Logical_Operations] → Conclusions
Tasks well suited for	• Stochastic reasoning tasks with multiple valid paths • Problems with high reasoning error rates • Mathematical problem solving with verification	• Formal logic problems • Structured reasoning tasks • Tasks requiring careful premise tracking
Implementation Considerations	• Computationally expensive • Requires effective aggregation strategy • Sampling temperature tuning critical	• Requires models trained on logical formalism • Structured format for logical operations • Limited to domains expressible in formal logic
Observations about performance	• Significantly outperforms single-path CoT • Reduces variance in performance • Robust against individual reasoning failures	• Superior performance on formally structured problems • Reduced hallucination in logical domains • Improved consistency in deductive reasoning
Prompt Acquisition	Manual	Manual
Prompt Turn	Single	Multi

2: Reasoning and Logic

Dimension	2.5 Chain-of-Symbol (CoS) Prompting	2.6 Tree-of-Thoughts (ToT) Prompting
What is it?	• Symbolic abstraction and manipulation • Intermediate symbolic representations • Domain-specific notation processing	• Breadth-first search over reasoning paths • Explicit state space exploration • Deliberate branching and evaluation
How it works?	• Symbolic transformation sequence: Problem → Symbolic Representation → Symbolic Operations → Solution	• Hierarchical tree structure: Root → [Branch_1, Branch_2, ...] → [Evaluation] → [Pruning] → Solution
Tasks well suited for	• Mathematical reasoning • Formal disciplines (chemistry, physics) • Programming and algorithm design	• Problems with decision points and backtracking • Planning and search problems • Complex games and puzzles
Implementation Considerations	• Requires domain expertise • Symbol parsing and generation capabilities • Context window must support symbolic representation	• Exponential scaling with problem depth • Requires effective state evaluation heuristics • Complex implementation with state tracking
Observations about performance	• Superior performance in mathematics • Enables complex multi-step derivations • Provides concise intermediate representations	• Superior performance on search-based problems • Provides multiple solution paths with quality ranking
Prompt Acquisition	Manual	Retrieval Based
Prompt Turn	Multi	Multi

2: Reasoning and Logic

Dimension	2.7 Graph-of-Thought (GoT) Prompting	2.8 System2Attention Prompting
What is it?	• Non-linear interconnected reasoning • Multi-directional information flow • Node-edge relationship modeling	• Dual-system cognitive architecture • Fast intuitive processing with slow deliberate verification
How it works?	• Directed graph with required connections • Nodes = {N1, N2, ...} • Edges = {E_i,j}	• Feedback loop with verification: System1 → [Attention_Filter] → System2 → [Verification] → Output
Tasks well suited for	• Knowledge graph reasoning • Interdependent concept relationships • Complex systems analysis	• Problems requiring both intuition and verification • Tasks with systematic biases or errors • High-stakes decision making
Implementation Considerations	• Complex graph representation within prompts • Requires graph construction and traversal • Challenging to maintain coherent global state	• Requires metacognitive capability in the model • Complex attention allocation mechanism • Dual-phase processing increases complexity
Observations about performance	• Excels at interconnected reasoning problems • Supports cyclic reasoning • Enables complex relationship modeling	• Reduces systematic errors and biases • Improves performance on reasoning tasks • Enables focused computational resource allocation
Prompt Acquisition	Retrieval Based	Manual
Prompt Turn	Multi	Single

2: Reasoning and Logic

Dimension	2.9 Thread of Thought (ThoT) Prompting	2.10 Chain of Table Prompting
What is it?	• Interleaved reasoning and reflection • Continuous refinement through self-dialogue • Dynamic adjustment of reasoning strategy	• Structured tabular reasoning • Information organization in explicit tabular format • Systematic data transformation and analysis
How it works?	• Spiral progression with reflection points: Initial_Thought → Reflection_1 → Refined_Thought → Reflection_2 → ... → Output	• Table-mediated processing: Problem → [Table_Construction] → [Table_Operations] → [Table_Analysis] → Solution
Tasks well suited for	• Problems benefiting from iterative refinement • Tasks requiring error correction • Complex reasoning with potential dead ends	• Structured data analysis • Multi-variable tracking problems • Tasks benefiting from explicit organization
Implementation Considerations	• Requires models capable of self-criticism • Context window must accommodate history • Complex prompt structure with interleaved components	• Requires effective tabular formatting in text • Table size limited by context window • Complex table operations challenging to express
Observations about performance	• Progressive improvement in reasoning quality • Robust against initial reasoning errors • Enables course correction	• Superior performance on data-organization tasks • Enables systematic tracking of multiple variables
Prompt Acquisition	Hybrid	Manual
Prompt Turn	Multi	Multi

THANK YOU