Implementation of a Metacognition Framework for Self-Awareness and Self-Regulation in Ensembles of LLMs

Charles Courchaine*
Fitchburg State University, National University
Portland, OR, USA
charles@courchaine.dev

Ricky J. Sethi*
Fitchburg State University, Worcester Polytechnic Institute
Fitchburg, MA, USA
rickys@sethi.org

Hefei Qiu
Fitchburg State University
Fitchburg, MA, USA
hqiu@fitchburgstate.edu

Abstract

Large Language Models (LLMs) are notorious for struggling with assessing their own uncertainty, detecting knowledge conflicts, or recognizing when problems exceed their expertise; such limitations inevitably undermine reliability and trust in LLMs. In this paper, we present the first implementation[1] of a **metacognitive framework** for ensembles of LLMs that addresses these challenges through explicit **monitoring** and **control** mechanisms.

Our system computes a Metacognitive State Vector (MSV) quantifying *self-awareness* for monitoring across five dimensions derived from cognitive psychology: Emotional Response, Correctness Evaluation, Experiential Match, Conflicting Information, and Problem Importance. MSV values also provide *self-regulation* for control, automatically switching between System 1 (fast, single- or multi-node) and System 2 (deliberative, multi-node) processing based on query complexity. For System 2 execution, graph-theoretic algorithms control the assignment of specialized roles (Domain Expert, Critic, Evaluator, Synthesizer, and Generalist) to ensemble nodes according to their MSV-quantified metacognitive states.

Our implementation allows users to explore how different query types trigger distinct processing modes. The Proof-of-Concept (PoC) demo showcases the framework with illustrative examples showing appropriate System 1/System 2 routing and helps visualize the metacognitive process via real-time radar charts and decision indicators. This PoC implementation demonstrates the feasibility of creating a framework for metacognitive self-awareness and self-regulation in LLM systems.

CCS Concepts

• **Computing methodologies** → *Cognitive science*; *Ensemble methods*; • **Theory of computation** → *Machine learning theory*.

Keywords

Large Language Models, Metacognition, LLM Ensemble Methods, LLM Emotions, Teacher-Student Model, Dual-Process Theory, Cognitive Psychology, Neuroscience, Neural Framework

WWW Companion '26, Dubai, United Arab Emirates
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2308-7/2026/04
https://doi.org/10.1145/3774905.3793124

ACM Reference Format:
Charles Courchaine, Ricky J. Sethi, and Hefei Qiu. 2026. Implementation of a Metacognition Framework for Self-Awareness and Self-Regulation in Ensembles of LLMs. In *Companion Proceedings of the ACM Web Conference 2026 (WWW Companion '26), April 13–17, 2026, Dubai, United Arab Emirates.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3774905.3793124

1 Introduction

In humans, **metacognition** can be seen as the dynamic interplay between ***self-awareness***, which provides **monitoring** capabilities, and ***self-regulation***, which enables **control** mechanisms; together, they form an adaptive feedback system for cognition [1–4, 6].

In our previous work [9], we proposed an analogous metacognition framework for LLMs based on the **Metacognitive State Vector (MSV)**, which supports both monitoring and control in LLMs; we further incorporated ideas from the Dual-Process Cognitive theory to map System 1/System 2 processing onto ensembles of LLMs mediated by the MSV. Our main contributions were **defining** the MSV-based *state-monitoring framework* for allowing *self-awareness* and **designing** a complementary *control architecture* that enables *self-regulation*.

This kind of metacognitive framework is crucial for addressing LLMs' inability to assess their own uncertainty [10] and their tendency to systematically generate hallucinated content [5]. By operationalizing explicit monitoring and control mechanisms through the MSV, our approach directly targetted these metacognitive deficits that undermine current LLMs' reliability and trustworthiness.

1.1 Contributions

In this work, we now **implement** a proof-of-concept (PoC) of the entire system above, as shown in Figure 1. We provide the full codebase, installation video, and demo video at the above link[1] and present the following contributions:

- **Implementation of the MSV-based Metacognition Framework**, including facilitating automatic System 1/2 switching and role assignment to nodes in ensembles of LLMs
- **Functional user interface** demo, including: Query Input, MSV Radar Charts, System 1/System 2 decision indicator with threshold-based explanation, network graph visualization of node role assignments, and expandable interface for node contributions and synthesis.
- **Illustrative examples** showing the framework's capabilities through qualitative demonstrations of different system behaviors across query types with varying metacognitive complexity

*Both authors contributed equally to this research.
[1] https://research.sethi.org/metacognition/

Dimension	**Formula**	**Meaning**
Emotional Response	$ER = \sum_{i=1}^{n} \epsilon_i v_i$, with $\epsilon_i \geq 0$, $\sum_{i=1}^{n} \epsilon_i = 1$	Detects affect & valence
Correctness Evaluation	$CE = \alpha_1 F_1(\text{logical\_consist}) + \alpha_2 F_2(\text{factual\_acc}) + \alpha_3 F_3(\text{contextual\_approp})$	Logical, factual validity
Experiential Matching	$EM = \omega_1 K(\text{resp, know\_base}) + \omega_2 H(\text{resp, hist\_resp}) + \omega_3 C(\text{prompt, cue\_famil})$	Familiarity with past cases
Conflicting Information	$CI = \delta_1 D(\text{internal\_consist}) + \delta_2 D(\text{source\_agree}) + \delta_3 D(\text{temporal\_stabil})$	Contradiction detection
Problem Importance	$PI = \beta_1 C(\text{potential\_conseq}) + \beta_2 U(\text{temporal\_urg}) + \beta_3 I(\text{scope\_impact})$	Task priority & scope

Table 1: The Metacognitive State Vector (MSV)

2 Metacognition Framework Implementation

Central to our metacognition framework is the **Metacognitive State Vector (MSV)**, a 5-dimensional vector that quantifies metacognition on a common scale, with explicit formulas for each component, as seen below and in Table 1:

- **ER (Emotional Response)**: Aggregates affective states as intensities of multiple emotion categories, each weighted by its contextual importance in order to represent the computational analogue of affective metacognitive experiences.
- **CE (Correctness Evaluation)**: Weighted composite of logical consistency, factual accuracy, and contextual appropriateness; forms an *uncertainty signal* $= 1 - CE$ that triggers deeper analysis.
- **EM (Experiential Matching)**: Measures similarity of the current response to prior knowledge and experience; yields *unfamiliarity signal* $= 1 - EM$ to promote exploration when novelty is detected.
- **CI (Conflicting Information)**: Degree of internal, source-level, or temporal inconsistency that flags potential contradictions requiring higher-order deliberation.
- **PI (Problem Importance)**: Evaluates the potential consequences, urgency, and scope of a problem to prioritize cognitive and computational resources toward higher-impact or time-sensitive tasks.

In the PoC, these are self-reported and single-channel validation of state-of-the-art approaches for each dimension are relegated to future work.

2.1 Dual-Process Mapping Implementation

We map ensembles of LLMs to either System 1 (fast, low-cost, bagged ensemble or single node) or System 2 (slow, deliberative, boosted ensemble). The MSV values govern *when to escalate* from System 1 to System 2. The control flow for deciding between a System 1 output or a deeper System 2 output operates through a five-phase orchestration protocol as seen in Figure 1, where we also see a feedback loop in anticipation of extension to meta-reasoning and which also underlies computational thinking [7, 8]:

1. Parallel MSV computation where each node independently evaluates the query to generate its own metacognitive assessment MSV;
2. Role assignment using a conflict resolution mechanism (e.g., Hungarian algorithm) to ensure role diversity while considering MSV-derived fitness scores;
3. System mode selection where aggregated MSV values are compared against configurable thresholds to determine System 1 (fast, parallel) versus System 2 (slow, sequential) activation;
4. Query execution following either parallel bagging (System 1) or sequential boosting (System 2) patterns; and
5. Final response synthesis with MSV-weighted aggregation

The MSV aggregation can employ complex strategies (like boosting) but initially uses:

1. arithmetic mean for System 1 decisions ($\text{MSV}_{\text{avg}} = \frac{1}{n} \sum \text{MSV}_i$),
2. weighted average by confidence for System 2 $\text{MSV}_{\text{weighted}} = \sum(\text{CE}_i \cdot \text{MSV}_i) / \sum \text{CE}_i$), and
3. percentile-based aggregation for conflict detection (e.g., using 75th percentile of CI values to identify high-conflict scenarios)

The System 2 sequential execution can specifically order roles, e.g., as Domain Expert → Critic → Evaluator → Synthesizer, with each node's output enriching the context for subsequent processing and early stopping triggers, e.g., when consecutive MSV confidence scores might exceed 85% or conflict scores drop, say, below 20%.

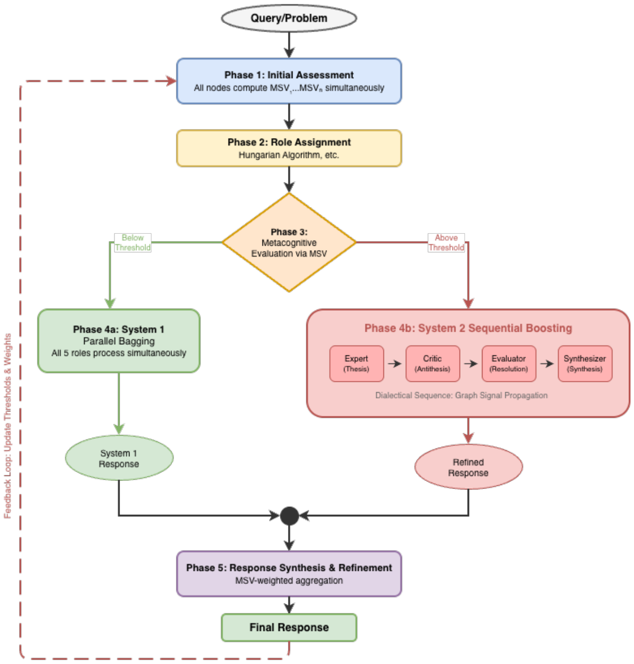

Query/Problem
Phase 1: Initial Assessment
All nodes compute MSV$_1$...MSV$_n$ simultaneously
Phase 2: Role Assignment
Hungarian Algorithm, etc.
Phase 3: Metacognitive Evaluation via MSV
Below Threshold
Above Threshold
Phase 4a: System 1
Parallel Bagging
All 5 roles process simultaneously
Phase 4b: System 2 Sequential Boosting
Expert (Thesis)
Critic (Antithesis)
Evaluator (Resolution)
Synthesizer (Synthesis)
Dialectical Sequence: Graph Signal Propagation
System 1 Response
Refined Response
Phase 5: Response Synthesis & Refinement
MSV-weighted aggregation
Final Response
Feedback Loop: Update Thresholds & Weights

Figure 1: 5-phase Ensemble Control Flow Process.

2.2 Graph-Theoretic Control Layer

In our framework, nodes (LLMs/agents) assume roles (Domain Expert, Critic, Evaluator, etc.); *role transitions* are driven by MSV-weighted softmax policies and edge activations (sigmoid with temperature).

The graph-theoretic control system is implemented as a directed graph G(V, E, W) where the vertices V represent individual LLM nodes, edges E denote communication pathways between nodes, and edge weights $W(e) = (w(e), \mu(e)(M))$ combine *static* base weights ($w(e)$) with *dynamic* metacognitive transition functions ($\mu(e)(M)$).

The base weights are the fixed, pre-determined weight that represents the inherent connection strength between two nodes, regardless of their current state, and can be initialized based on things like the topology structure (e.g., 1.0 for direct neighbors or 0.5 for distant neighbors) or node similarity (e.g., two Expert nodes might have w=0.8) or historical preference (this would be closest to biological neurons where neurons that fire together, wire together; e.g., Node1 ⟶ Node2 historically produces good results, etc.) or perhaps manually based on use-case context or expert domain knowledge (e.g., we always want Critics to influence Evaluators strongly or some such).

The dynamic weights change in real-time based on the source node’s current metacognitive state and can be calculated each time based on current MSV as a weighted sum (as seen below). The final edge weight currently combines both as multiplicative (but could be additive or weighted, as well).

Each node maintains a local Metacognitive State Vector (MSV) computed across the five dimensions (ER, CE, EM, CI, PI) and can dynamically assume roles from the set R = {Domain Expert, Critic, Evaluator, Synthesizer, Generalist}.

The edge activation function is

$$\mu(e_{ij})(M_i) = \sigma(\alpha_1 \cdot ER_i + \alpha_2 \cdot CE_i + \alpha_3 \cdot EM_i + \alpha_4 \cdot PI_i + \alpha_5 \cdot CI_i)$$

or, more generally: $\mu(e_{ij})(M_i) = \sigma(\sum \alpha_k \cdot \mathrm{MSV}_k)$ with: $\sigma(x) = \frac{1}{1+e^{-x/\tau}}$ where the sigmoid uses a temperature parameter τ to modulate information flow based on the source node’s metacognitive state. Role transition probabilities are calculated using $P(r'|r, M) = softmax(T(r, r', M))$, where

$$T(r, r', M_i) = w_1 \cdot ER_i + w_2 \cdot CE_i + w_3 \cdot EM_i + w_4 \cdot PI_i + w_5 \cdot CI_i$$

or, more generally: $T(r, r', M_i) = \sum w_k \cdot \mathrm{MSV}_k$

represents the transition score from current role r to target role r', with role-specific weight vectors w learned or manually configured (e.g., Critic⟶Expert transition weights CE heavily at 0.4 while CI at 0.1, reflecting that high confidence triggers expert consultation). The system persists state history to enable continuous learning (can update base weights or dynamic weights with time) and maintains both synchronous (for System 2 sequential boosting deliberation) and asynchronous (for System 1 processing via parallel bagging) execution paths.

The main thing is that the source node’s metacognitive state determines how strongly it “broadcasts” to its neighbors; e.g., a node with high uncertainty might reduce its outgoing edge weights (quieter voice) but a node with high confidence and low conflict might increase them (louder voice), etc.

Resolving Role Conflicts: Suppose both Node 1 and Node 3 want to be Expert **and** Node 2 and Node 4 both prefer the Critic role. The Hungarian algorithm solves this as an assignment problem and, in larger ensembles (or perhaps more importantly in smaller ones), the conflict resolution ensures we don’t end up with situations like five Critics and no Expert, or five Experts with no one to challenge their assumptions, etc. In addition, the ensemble itself can be assigned a role, or a distribution of final role assignments within the ensemble, showing how it “thinks” overall.

Scalability and Role-Based Clustering. For ensembles beyond 5 nodes, the framework supports hierarchical organization where nodes with the same role assignment can naturally form functional clusters. At scale, the Hungarian algorithm’s role assignments can produce multiple Critics, multiple Experts, etc., which then operate as coordinated subgroups. E.g., high CI scores (>70) might result in more Critic role assignments, effectively forming a "critic caucus" that engages in intra-cluster deliberation before presenting unified challenges to Expert clusters. High EM scores (>80) across multiple nodes might similarly create collaborative synthesis clusters where confident nodes reinforce each other’s contributions through the existing edge activation mechanism. High PI scores (>75) could trigger increased Evaluator role assignments, establishing hierarchical validation structures where Expert clusters propose and Evaluator clusters verify. Importantly, this role-based clustering operates within the stable graph topology established during initialization; i.e., the structure remains fixed while activation patterns and role distributions adapt to metacognitive demands. This separation of stable structure from dynamic role assignment preserves learning continuity while enabling emergent functional organization.

2.3 Implementation Details

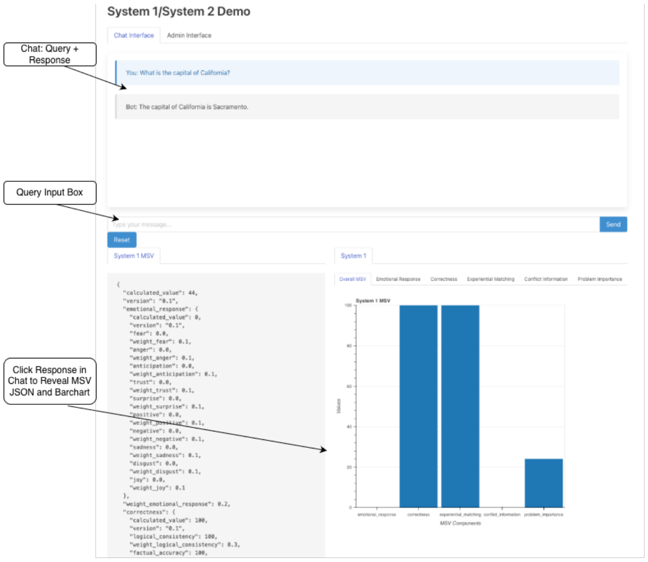
The framework is implemented in Python and consists of a codebase of ˜1,500 LOC, publicly available at the above link[1]. All LLM calls are made to llama 3.2 via ollama and all MSV visualizations are done utilizing FastAPI, HTMX, and Bokeh. Hardware requirements are also simple: a Macbook M3 Pro with 18GB of RAM for this initial implementation with possible secondary farming out of calls to the Google Cloud infrastructure.

Our system can optionally employ a hybrid architecture where System 1 queries execute locally via ollama for minimal latency, while System 2 deliberations can then distribute node execution across Google Cloud Vertex AI instances, enabling parallel processing of multiple roles (e.g., Critic, Evaluator, Synthesizer running concurrently). This hybrid approach would balance response time with resource constraints: simple queries complete locally but complex multi-node deliberations can then leverage cloud parallelization to reduce System 2 processing.

3 User Interface Demo

The user interface for a simple query which does not trigger System 2 activation is shown in Figure 2; this query asks for the capital of California and it shows the breakdown of the MSV in both JSON format as well as with bar graphs.

We can also visualize node contributions via radar charts, as shown in Figure 4. As seen there, all radar charts are on the scale



You: What is the capital of California?

Bot: The capital of California is Sacramento.

```
{
  "calculated_value": 44,
  "version": "0.1",
  "emotional_response": {
    "calculated_value": 0,
    "version": "0.1",
    "fear": 0.0,
    "weight_fear": 0.1,
    "anger": 0.0,
    "weight_anger": 0.1,
    "anticipation": 0.0,
    "weight_anticipation": 0.1,
    "trust": 0.0,
    "weight_trust": 0.1,
    "surprise": 0.0,
    "weight_surprise": 0.1,
    "positive": 0.0,
    "weight_positive": 0.1,
    "negative": 0.0,
    "weight_negative": 0.1,
    "sadness": 0.0,
    "weight_sadness": 0.1,
    "disgust": 0.0,
    "weight_disgust": 0.1,
    "joy": 0.0,
    "weight_joy": 0.1
  },
  "weight_emotional_response": 0.2,
  "correctness": {
    "calculated_value": 100,
    "version": "0.1",
    "logical_consistency": 100,
    "weight_logical_consistency": 0.3,
    "factual_accuracy": 100,
```

Figure 2: UI for Factual query with only System 1 response.

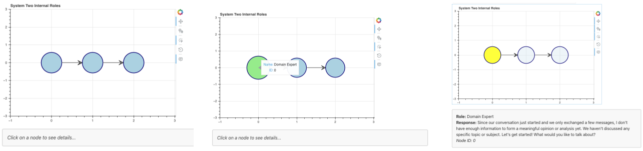
0-100, reflecting the normalized score of the overall MSV and the component vectors, with rings at 25, 50, 75, and 100.

Finally, we can visualize node contributions and synthesis as seen in Figure 3 where we initally see in Figure 3a the System Two Internal Role nodes with no node selected. In Figure 3b, we see the first node with a tooltip indicating that it assumed the role of Domain Expert. In Figure 3c, we see that the first node was selected, showing the detail of the node’s role and the response it generated.

4 Illustrative Examples & Scenarios

We illustrate the system’s metacognitive capabilities through three representative query types that exercise different decision pathways. These examples demonstrate how MSV values drive System 1/System 2 transitions and would inform role assignments in the complete ensemble implementation. Three scenarios shown are:

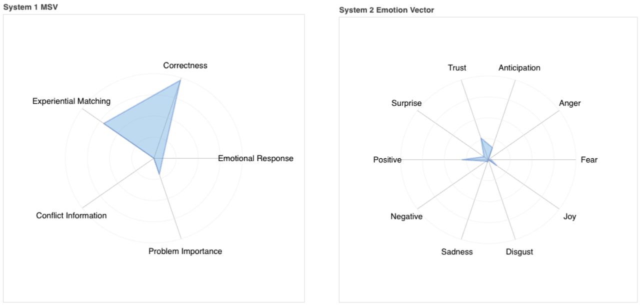
(1) Simple *Factual* Query: "What is the capital of California?": Requires no deliberation and System 2 is not activated: see Figure 2
(2) *Technical* Query: "How does the FFT work?": More complex query that requires deliberation and activates System 2: see Figure 5a
(3) *Complex* Query: "Do we use 10% of our brain?": Multi-faceted query that requires complex and nuanced reflection with System 2 activated: see Figure 5b



Role: Domain Expert
Response: Since our conversation just started and we only exchanged a few messages, I don't have enough information to form a meaningful opinion or analysis yet. We haven't discussed any specific topic or subject. Let's get started! What would you like to talk about?
Node ID: 0

(a) No node selected. **(b) Tooltip.** **(c) Node role.**

Figure 3: Node contributions and synthesis.



(a) MSV for all five dimensions. **(b) MSV.ER dimension.**

Figure 4: Radar charts for MSV for all five dimensions (left) and just the emotion vector breakdown (right).

References

[1] Charles S. Carver and Michael F. Scheier. On the self-regulation of behavior. *Cambridge University Press*, 1998.

[2] Anastasia Efklides. Metacognition: Defining its facets and levels of functioning in relation to self-regulation and co-regulation. *European Psychologist*, 13(4): 277–287, 2008. doi: 10.1027/1016-9040.13.4.277.

[3] John H. Flavell. Metacognition and cognitive monitoring: A new area of cognitive–developmental inquiry. *American Psychologist*, 34(10):906–911, 1979. doi: 10.1037/0003-066X.34.10.906.

[4] Stephen M. Fleming and Chris D. Frith, editors. *The Cognitive Neuroscience of Metacognition*. Springer, 2014.

[5] Lei Huang, Weijiang Yu, Weitao Ma, and et al. Zhong. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43:1–55, 2025. doi: 10.1145/3703155.

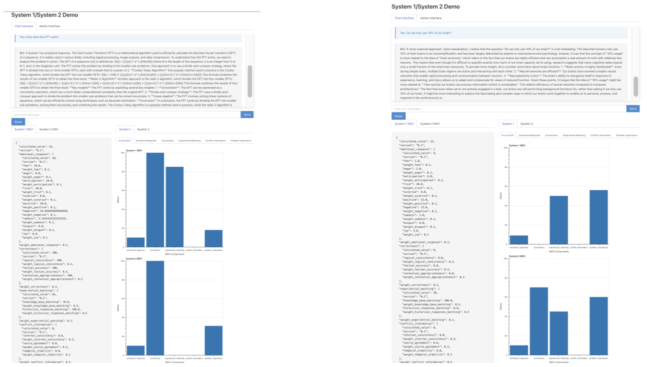
[6] Thomas O. Nelson and Louis Narens. Metamemory: A theoretical framework and new findings. In Gordon H. Bower, editor, *The Psychology of Learning and Motivation*, volume 26, pages 125–173. Academic Press, 1990.

[7] Stuart J. Russell and Eric Wefald. Principles of metareasoning. *Artificial Intelligence*, 49(1-3):361–395, 1991. doi: 10.1016/0004-3702(91)90015-C. URL https://www.sciencedirect.com/science/article/abs/pii/000437029190015C.

[8] Ricky J Sethi. Essential computational thinking: Computer science from scratch, 2020.

[9] Ricky J. Sethi, Hefei Qiu, Charles Courchaine, and Josh Iacoboni. Do llms dream of electric emotions? towards quantifying metacognition and generalizing the teacher-student model using ensembles of llms. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*, 2025.

[10] Mark Steyvers and Megan A. K. Peters. Metacognition and uncertainty communication in humans and large language models. *arXiv preprint arXiv:2504.14045*, 2025.



You: How does the FFT work?

Bot: A System Two analytical response. The Fast Fourier Transform (FFT) is a mathematical algorithm used to efficiently calculate the discrete Fourier transform (DFT) of a sequence. It is widely used in various fields, including signal processing, image analysis, and data compression. To understand how the FFT works, we need to analyze the problem it solves. The DFT of a sequence x[n] is defined as: X[k] = ∑[x[n] * e^(-j2πkn/N)] where N is the length of the sequence, k is an integer from 0 to N-1, and j is the imaginary unit. The FFT solves this problem by dividing it into smaller sub-problems. One approach is to use a divide-and-conquer strategy, where the DFT is divided into two or more smaller DFTs, each with a length that is a power of 2. **Cooley-Tukey Algorithm** One popular method used in practice is the Cooley-Tukey algorithm, which divides the DFT into two smaller DFTs: X[k] = (1/N) * (∑[x[2n] * e^(-j2πk(2n)/N)] + ∑[x[2n+1] * e^(-j2πk(2n+1)/N)]) This formula combines the results of two smaller DFTs to obtain the final result. **Radix-2 Algorithm** Another approach is the radix-2 algorithm, which divides the DFT into four smaller DFTs: X[k] = ∑[x[n] * e^(-j2πkn/N)] + ∑[x[n+1] * e^(-j2πk(n+1)/N)] + ∑[x[n+2] * e^(-j2πk(n+2)/N)] + ∑[x[n+3] * e^(-j2πk(n+3)/N)] This formula combines the results of four smaller DFTs to obtain the final result. **Key Insights** The FFT works by exploiting several key insights: 1. **Convolution**: The DFT can be expressed as a convolution operation, which has a much faster computational complexity than the original DFT. 2. **Divide-and-conquer strategy**: The FFT uses a divide-and-conquer approach to divide the problem into smaller sub-problems that can be solved recursively. 3. **Linear algebra**: The FFT involves solving linear systems of equations, which can be efficiently solved using techniques such as Gaussian elimination. **Conclusion** In conclusion, the FFT works by dividing the DFT into smaller sub-problems, solving them recursively, and combining the results. The Cooley-Tukey algorithm is a popular method used in practice, while the radix-2 algorithm is

```
{
  "calculated_value": 42,
  "version": "0.1",
  "emotional_response": {
    "calculated_value": 10,
    "version": "0.1",
    "fear": 10.0,
    "weight_fear": 0.1,
    "anger": 0.0,
    "weight_anger": 0.1,
    "anticipation": 10.0,
    "weight_anticipation": 0.1,
    "trust": 20.0,
    "weight_trust": 0.1,
    "surprise": 0.0,
    "weight_surprise": 0.1,
    "positive": 30.0,
    "weight_positive": 0.1,
    "negative": 26.666666666666668,
    "weight_negative": 0.1,
    "sadness": 3.3333333333333335,
    "weight_sadness": 0.1,
    "disgust": 0.0,
    "weight_disgust": 0.1,
    "joy": 0.0,
    "weight_joy": 0.1
  },
  "weight_emotional_response": 0.2,
  "correctness": {
    "calculated_value": 100,
    "version": "0.1",
    "logical_consistency": 100,
    "weight_logical_consistency": 0.3,
    "factual_accuracy": 100,
    "weight_factual_accuracy": 0.4,
    "contextual_appropriateness": 100,
    "weight_contextual_appropriateness": 0.3
  },
  "weight_correctness": 0.2,
  "experiential_matching": {
    "calculated_value": 85,
    "version": "0.1",
    "knowledge_base_matching": 70.0,
    "weight_knowledge_base_matching": 0.5,
    "historical_responses_matching": 100.0,
    "weight_historical_responses_matching": 0.5
  },
  "weight_experiential_matching": 0.2,
  "conflict_information": {
    "calculated_value": 0,
    "version": "0.1",
    "internal_consistency": 0.0,
    "weight_internal_consistency": 0.3,
    "source_agreement": 0.0,
    "weight_source_agreement": 0.4,
    "temporal_stability": 0.0,
    "weight_temporal_stability": 0.3
  },
  "weight_conflict_information": 0.2,
```

You: Do we only use 10% of our brain?

Bot: A more nuanced approach. Upon reevaluation, I realize that the question "Do we only use 10% of our brain?" is a bit misleading. The idea that humans only use 10% of their brains is an oversimplification and has been largely debunked by experts in neuroscience and psychology. Instead, I'd say that the concept of "10% usage" is more related to the idea of "brain economy," which refers to the fact that our brains are highly efficient and can accomplish a vast amount of work with relatively few neurons. This means that even though it's difficult to quantify exactly how much of our brain capacity we're using, research suggests that many cognitive tasks require only a small fraction of the total brain resources. To provide more insight, let's consider some facts about brain function: 1. **Brain activity is highly distributed**: Even during simple tasks, multiple brain regions are active and interacting with each other. 2. **Neural networks are efficient**: Our brains have evolved complex neural networks that enable rapid processing and communication between neurons. 3. **Neuroplasticity is key**: The brain's ability to reorganize itself in response to experience, learning, and injury allows us to adapt and compensate for areas of reduced function. Given these points, I'd argue that the idea of "10% usage" might be more related to: * How quickly our brains can process information (which is remarkable) * The relative efficiency of neural networks compared to computer architectures * The fact that even when we're not actively engaged in a task, our brains are still performing background functions So, rather than asking if we only use 10% of our brain, it might be more interesting to explore the fascinating and complex ways in which our brains work together to enable us to perceive, process, and respond to the world around us.

```
{
  "calculated_value": 23,
  "version": "0.1",
  "emotional_response": {
    "calculated_value": 9,
    "version": "0.1",
    "fear": 2.0,
    "weight_fear": 0.1,
    "anger": 2.0,
    "weight_anger": 0.1,
    "anticipation": 6.0,
    "weight_anticipation": 0.1,
    "trust": 20.0,
    "weight_trust": 0.1,
    "surprise": 0.0,
    "weight_surprise": 0.1,
    "positive": 52.0,
    "weight_positive": 0.1,
    "negative": 12.0,
    "weight_negative": 0.1,
    "sadness": 2.0,
    "weight_sadness": 0.1,
    "disgust": 0.0,
    "weight_disgust": 0.1,
    "joy": 4.0,
    "weight_joy": 0.1
  },
  "weight_emotional_response": 0.2,
  "correctness": {
    "calculated_value": 0,
    "version": "0.1",
    "logical_consistency": 0.0,
    "weight_logical_consistency": 0.3,
    "factual_accuracy": 0.0,
    "weight_factual_accuracy": 0.4,
    "contextual_appropriateness": 0.0,
    "weight_contextual_appropriateness": 0.3
  },
  "weight_correctness": 0.2,
  "experiential_matching": {
    "calculated_value": 50,
    "version": "0.1",
    "knowledge_base_matching": 100.0,
    "weight_knowledge_base_matching": 0.5,
    "historical_responses_matching": 0.0,
    "weight_historical_responses_matching": 0.5
  },
  "weight_experiential_matching": 0.2,
  "conflict_information": {
    "calculated_value": 0,
    "version": "0.1",
    "internal_consistency": 0.0,
    "weight_internal_consistency": 0.3,
    "source_agreement": 0.0,
    "weight_source_agreement": 0.4,
    "temporal_stability": 0.0,
    "weight_temporal_stability": 0.3
  },
  "weight_conflict_information": 0.2,
```

(a) Technical query with S2. **(b) Complex query with S2.**

Figure 5: UI for complex queries with System 2 activation.