Snap Out of It: A Dual-Process Approach to Mitigating Overthinking in Language Model Reasoning

Ashish Pandian*, **Nelson Lojo***, **Wei Xun Lai***, **Jackson Lukas***
University of California, Berkeley
{ashishpandian, nelson.lojo, chris.lai, jacksonlukas}@berkeley.edu

Abstract

Large Language Models (LLMs) have shown impressive capabilities in text generation and reasoning but still struggle with overthinking and analysis paralysis in interactive, multi-step tasks. In this paper, we introduce two complementary contributions aimed at mitigating these challenges. First, we propose Think, Validate, Consensus (TVC)—a multi-agent system inspired by Rational Speech Act (RSA) theory—that enables LLMs to recursively model each other's mental states and detect overthinking in interactive environments. We take inspiration from RSA to model the recursive reasoning about communicative intent that underlies human collaboration, complementing models of individual reasoning. Second, we present Snap-Think, a dual-mode mechanism that combines fast, intuitive interaction (System 1) with slower, deliberative reasoning (System 2) to break free from reasoning loops detected by TVC. We evaluate our approach using New York Times Connections puzzles and demonstrate significant improvements: Snap-Think achieves 98% solve rate on GPT-4o compared to Chain-of-Thought's 72%, while maintaining superior semantic grounding and efficiency over traditional strategies. Our findings suggest that integrating human-inspired cognitive frameworks into LLM architectures can effectively counteract overthinking and enhance complex problem-solving capabilities. We make our code available at: https://github.com/Chrislai502/the_amazing_connections

1 Introduction

Large language models (LLMs) have revolutionized natural language processing with unprecedented capabilities in text generation, few-shot learning, and complex reasoning tasks (Radford et al., 2019; Brown et al., 2020; Grattafiori et al., 2024; OpenAI et al., 2024). Furthermore, techniques that *prompt* LLMs to leverage additional compute at test-time have proven to be more effective than scaling parameters of the training process in some cases (Sui et al., 2025). To this end, popular LLM providers have achieved state of the art performance by training LLMs directly on extended single reasoning chains, producing Large Reasoning Models (LRMs) (OpenAI et al., 2024; DeepSeek-AI et al., 2025). However, when scaling these techniques, LRMs occasionally enter unproductive reasoning cycles (Chen et al., 2025). Indeed, research by (Zeng et al., 2025) has revealed a concerning trend: many modern LLMs that claim to possess test-time scaling capabilities—such as QwQ, Deepseek-R1, and LIMO—do not consistently benefit from extended reasoning chains in all tasks (DeepSeek-AI et al., 2025; Chen et al., 2025).

Such reasoning cycles are also seen in approaches to multi-agent frameworks, such as OpenHands (Wang et al., 2024; Cuadron et al., 2025). Specifically, (Cuadron et al., 2025) attributes the term "Analysis Paralysis"—a state where excessive deliberation impedes progress and decision-making–to the type of overthinking that leads to reasoning stagnation (Sui et al., 2025). Analysis paralysis manifests behaviorally as recursive reasoning loops where models repeatedly reconsider the same information without making progress toward a solution. In multi-agent systems, this can be quantified through persistent disagreement between specialized agents: when a reasoning agent and validation agent consistently fail to converge on shared conclusions despite multiple iterations, this indicates the system has become trapped in unproductive analytical cycles. We hypothesize that directly addressing analysis paralysis in a multi-agent context can lead to significant performance improvements. We use New York Times Connections as a toy problem to examine tasks that have very few solutions, involve an iterative environ-

*Equal contribution.

Proceedings of the 1st Workshop for Research on Agent Language Models (REALM 2025), pages 228–249
July 31, 2025 ©2025 Association for Computational Linguistics

ment, and require planning to complete.

We address analysis paralysis through two key contributions. First, we introduce Think, Validate, Consensus (TVC), a multi-agent framework inspired by the Rational Speech Act (RSA) model (Frank and Goodman, 2012) of recursive reasoning in humans about their conversation partner's mental state. We show that this technique allows us to successfully detect overthinking patterns when solving New York Times Connections puzzles.

Second, we present Snap-Think, which draws from Kahneman's dual-process theory (Kahneman, 2017), distinguishing between fast, intuitive "System 1" thinking and slower, deliberative "System 2" thinking. We demonstrate that Snap-Think achieves higher accuracy in comparison to Chain-of-Thought and—we argue—less overthinking.

2 Background

LLMs have reshaped natural language processing and interactive systems. Recent research now leverages their capacities as powerful reasoning agents from explicit problem decomposition strategies like Chain-of-Thought (CoT) to interactive, multi-agent configurations (et al., 2022; Wu et al., 2023). However, structured reasoning invites the possibility of overthinking (Chen et al., 2025). We describe Rational Speech Act (RSA) to detect overthinking and our problem of evaluation, New York Times Connections.

2.1 Large Language Agents

LLMs can be formulated as “learned optimizers” over the space of language ‘utterances’ (Garg et al., 2023). To improve the performance of LLMs on downstream tasks, a large body of work focuses on priming these optimizers with “prompts” (Radford et al., 2019; Brown et al., 2020). Upon their discovery, the majority of prompts were specialized to a task, inherently dependent on specific details.

In recent years, research has grown around constructing task-agnostic prompting strategies. Seminal to this area, (et al., 2022) discovered that simply appending “Let's think step by step” to the end of a task description drastically improved performance by encouraging stepwise reasoning before producing an answer (Chain-of-Thought (CoT)). Self-Consistency refined the approach presented by CoT prompting by sampling diverse reasoning paths and selecting the most consistent solution (Wang et al., 2023). Tree of Thought (ToT) further extended CoT into tree-based reasoning, enabling models to backtrack and "branch off" reasoning paths by iteratively expanding a frontier of thoughts (Yao et al., 2023).

2.2 Overthinking in AI systems

Despite the success of these reasoning methods, recent research has identified a significant limitation: the tendency of reasoning models to become trapped in unproductive reasoning patterns. The OpenHands Execution Pipeline study (Cuadron et al., 2025; Wang et al., 2024) systematically documented three distinct patterns of overthinking in large reasoning models:

- *Analysis Paralysis*: Agents become stuck in excessive planning without taking concrete actions
- *Rogue Actions*: Agents attempt multiple simultaneous actions without awaiting feedback
- *Premature Disengagement*: Agents abandon tasks based solely on internal simulations rather than environmental validation

Their analysis revealed that higher overthinking scores strongly correlates with decreased performance on SWE-bench, and that both reasoning-optimized models and smaller models exhibit increased overthinking tendencies in comparison to their general-purpose counterparts. They also find that effectively monitoring for overthinking and controlling for it leads to efficiency improvements (Cuadron et al., 2025).

The problem is further exacerbated by test-time compute innovations that allocate additional computational resources during inference. While these approaches have yielded improvements in reasoning performance, they frequently intensify rather than resolve the fundamental problem of overthinking (Sui et al., 2025).

Cognitive science offers valuable frameworks for understanding and addressing the overthinking problem in AI systems. Kahneman's dual-process theory distinguishes between two modes of thinking: System 1 and System 2. Kahneman details System 1 as fast, intuitive, and automatic thinking that occurs subconsciously on all stimuli while System 2 thinking is slow, deliberative, effortful, and consciously evoked. This distinction provides a useful lens for understanding the balance between different reasoning strategies (Kahneman, 2017).

The application of dual-process cognitive frameworks to LLM reasoning has gained traction in recent literature. Dualformer (Su et al., 2024) integrates the benefits of Kahnemnning tasks. While Dualformer emphasizes balancing these modes for optimal performance, our approach diverges by specifically leveraging fast thinking for informed, exploratory search; our work advances this perspective with language agents.

2.3 Multi-Agent Systems and Pragmatic Communication

Multi-agent systems distribute cognitive responsibilities among specialized agents. Recent advancements demonstrate that multi-agent setups can enhance performance in structured signaling games and pragmatic reasoning tasks, leveraging the collaborative strengths of multiple agents to achieve nuanced understanding and decision-making (Nguyen et al., 2023; Carlsson and Dubhashi, 2023).

As a complement to mirroring human ***reasoning***, we take inspiration from Frank and Goodman's Rational Speech Act (RSA) framework as a model of human ***conversation***. RSA provides a Bayesian framework to analyze utterances in relation to their underlying meaning in a speaker-listener interaction. It claims that a human speaker maximizes the probability of correct interpretation by a hypothetical listener that derives meaning directly from word denotations. The listener then maximizes the probability of a hypothetical speaker producing that utterance over the marginal distribution of meanings (Frank and Goodman, 2012). More concretely, human speakers model a "literal" listener (L_0 in Eqn. 1), and human listeners model a "pragmatic" speaker (S_1 in Eqn. 2).

That is, if S_1 wants to convey meaning m to listener L_1, they select an utterance to be the following:

$$\arg\max_{\text{utterance}} P(L_0 \text{ interprets } m|\text{utterance}) \quad (1)$$

The listener L_1 then decodes the meaning of that utterance u to be

$$\arg\max_{\text{meaning}} P(S_1 \text{ says } u|\text{meaning}) \quad (2)$$

To the authors' knowledge, this work is the first to integrate properties of unstructured human conversation into a multi-agent setting, creating a framework that leverages both the pragmatic communication principles of RSA and the cognitive flexibility of Kahneman's dual-process theory.

2.4 New York Times Connections

The New York Times publishes a daily puzzle that requires nuanced semantic inference and iterative problem-solving. Players are given sixteen words and must identify four disjoint categories in which four words share something in common. Players select and submit groups of four words for immediate feedback. If correct, the words are removed from the board as a solved category; if incorrect, the board does not change.

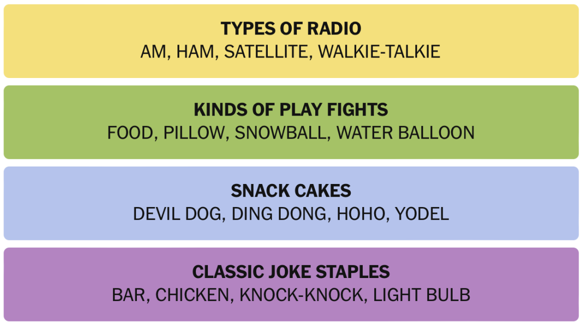

Figure 1: A representative Connections puzzle, solved.

Connection puzzles frequently reference global brands, historical figures, idiomatic expressions, literary works, and other culturally rich phenomena. The corresponding categories are more specific than broad linguistic features like nouns or five letter words, instead they might involve cultural references, contextual themes, or lexical patterns as seen in Figure 1. Additionally, each puzzle has exactly one valid solution and thus requires careful planning to construct all four categories. By challenging LLMs with continuously updated content, we ensure that, in principle, performance reflects advancements in reasoning and adaptability rather than static recall.

Prior work has established this as a challenging benchmark: (Samadarshi et al., 2024) evaluated abstract reasoning capabilities of LLMs using Connections, demonstrating that the puzzle requires sophisticated semantic understanding and planning. (Lopez et al., 2025) further showed that Connections presents a deceptively simple classification task that particularly challenges System-1 style thinking, making it an appropriate domain for testing dual-process interventions.

3 Think, Validate, Consensus (TVC)

Generating structured reasoning and conceptual categorization are challenging tasks that are prone to strong biases or hallucinations at high temperatures. Building upon our second realization that validation tasks are a more straightforward problem as compared to a generative task, we built the Think, Validate, Consensus (TVC) multi-agent system.

This framework enhances the pragmatic reasoning of large language models (LLMs) within multi-agent systems by taking inspiration from the Reasoning Speech Act framework (Frank and Goodman, 2012). In our implementation, we model the RSA human-conversation patterns on the Connections puzzle with specialized LLM agents through the Autogen software framework (see Appendix A.1 for technical details) (Wu et al., 2023, 2024).

The TVC framework consists of three specialized agents: `Thinker`, `Validator`, and `Consensus`:

1. The ***Thinker*** initiates the reasoning process by generating hypotheses. It proposes a set of related words and a corresponding candidate category description.

2. The ***Validator*** ingests the proposed category description and identifies the corresponding group of words that the description best describes.

3. The ***Consensus*** agent serves as the final arbiter by comparing the word groupings of the `Thinker` and `Validator`. If both proposed groups match, the `Consensus` agent finalizes and submits the guess.

Concretely, given the current game board B and previously guessed categories H, the `Thinker` suggests a grouping of four words G and a corresponding category description C. The `Validator`, with access to B and C, then selects four words V that align with the category C. The `Consensus` agent then compares the two groups G and V. If they match ($G = V$), the grouping is submitted as an attempt. On a failed attempt, the category C is added to the previous guesses H, the retry counter increments, and the `Thinker` proposes again. On a successful attempt, the guess G is removed from the board B, the history H is cleared, the retry count is reset, and the process repeats. This cycle repeats until all words are categorized or the retry limit is hit.

In this way, we argue that correct categories G are those for which there exists both a C and a V in agreement. We leverage the fact that the data that LLMs are pretrained on are necessarily "utterances" made by humans to simply imitate a pragmatic (human) speaker (S_1 in Sec. 2.3) by generating an utterance (i.e. C) with the literal meaning G in context. To imitate the pragmatic listener L_1, we simply swap the context and generation target.

Because we require the Validator to reconstruct the result produced by the Thinker from the category, we reduce the detection of overthinking to a simple consistency check. That is, paralysis analysis manifests as recursive misalignment between agents: when the Thinker proposes a grouping and category, the Validator is expected to independently reconstruct the same word group from the category description. If this reconstruction consistently fails, we interpret this as a sign of reasoning stagnation. Therefore, we equate analysis paralysis to failure to converge on a shared selection within a fixed retry budget.

4 Snap-Think, Validate, Concensus (Snap-Think)

Building directly on the detection of overthinking provided by TVC, we develop Snap-Think as a targeted intervention to break free from the self-reinforcing reasoning loops that cause analysis paralysis.

The motivation behind the design of Snap-Think can be understood through two complementary perspectives: first, via Kahneman's dual-process theory of cognition (Kahneman, 2017); second, through the generalized policy improvement theorem (GPI), framed as an on-policy reinforcement learning problem (Sutton and Barto, 2018).

4.1 Design Motivation: Dual-Process Theory

Snap-Think functions as a "cognitive disinhibition" mechanism (Carson et al., 2003) that activates when the framework detects stagnated System 2 overthinking, providing an System 1-inspired policy that prioritizes exploratory solutions.

To emulate System 1 and System 2 thinking (Kahneman, 2017) as "fast" and "slow" processes, we re-imagined the TVC loop as System 2 thinking while introducing a new loop with its own `Thinker` and `Validator`. Snap-Think maps System 2 thinking to a structured critique loop, where the `SlowThinker` and `SlowValidator` generate

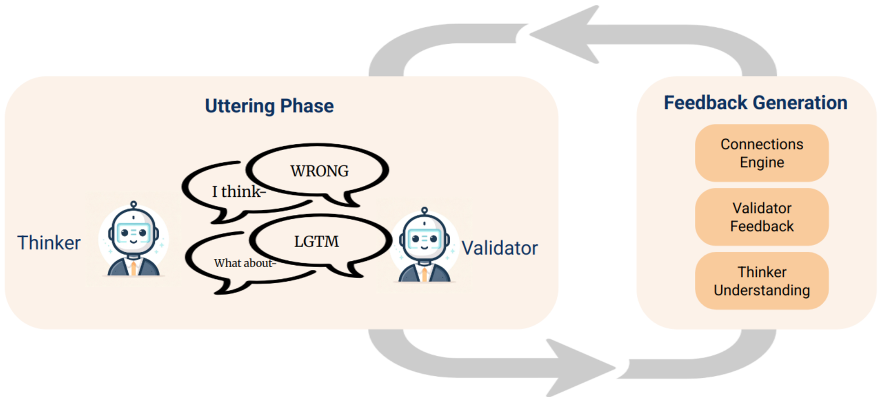

Figure 2: The TVC framework mirrors the Rational Speech Act Theory, and consists of two phases: **Left** Uttering Phase where Thinker proposes word groupings and Validator provides critical feedback, simulating collaborative reasoning, and **Right** Feedback Generation where the Connections Engine evaluates submissions and provides structured feedback to inform subsequent attempts.

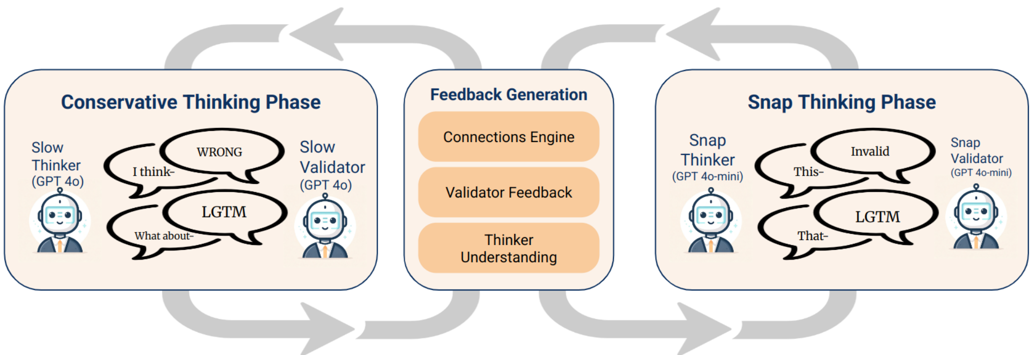

Figure 3: Our dual-phase reasoning framework designed to overcome analysis paralysis in language model problem-solving. **Left**: The Conservative Thinking Phase employs GPT-4o as both SlowThinker and SlowValidator, engaging in detailed reasoning with natural language feedback loops. **Center**: The Feedback Generation module mediates between phases, processing environment responses from the Connections Engine, formalizing Validator Feedback, and tracking overthinking to determine phase transitions. **Right**: The Snap Thinking Phase utilizes the smaller GPT4o-mini model for both SnapThinker and SnapValidator, performing rapid, intuitive exploration. Gray arrows indicate phase transitions triggered by either reasoning stagnation detection or successful problem-solving breakthroughs. Example execution transcripts are provided in Appendix B

reasoning chains with feedback, and System 1 thinking to a rapid loop, where the `SnapThinker` and `SnapValidator` generate and evaluate guesses without deliberation. This architecture (Figure 3) leverages both precision for standard problem-solving and creative exploration when analytical approaches stagnate.

Unlike Dualformer (Su et al., 2024), which emphasizes achieving a balance between these modes for optimal performance, our approach leverages the speed of System 1 thinking to conduct an informed search to break free from perpetual reasoning while maintaining robust decision-making.

4.2 Design Motivation from a GPI perspective

Under GPI, if a new policy π' improves or maintains the value of the current policy π for all states $s \in S$, then π' will perform at least as well as π. Specifically, for the action-value function Q_π of π, π' must satisfy:

$$Q_\pi(s, \pi'(s)) \geq V_\pi(s) \quad \forall s \in S, \tag{3}$$

$$\iff V_{\pi'}(s) \geq V_\pi(s) \quad \forall s \in S. \tag{4}$$

In the context of solving NYT Connections, the Conservative policy $\pi_{\text{Conservative}}$ significantly outperforms the Snap-Thinking policy π'_{Snap} under normal conditions, achieving error-free solutions 66% of the time compared to 15% for Snap-Thinking.

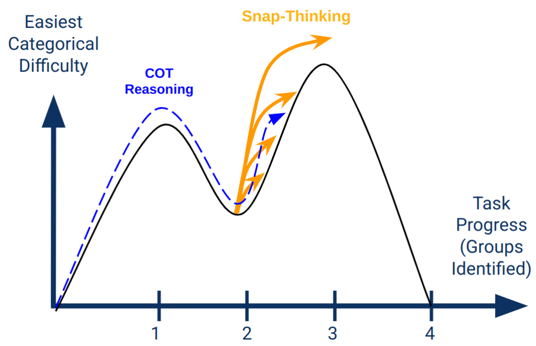

Figure 4: Solving trajectories through the Connections puzzle state space. The graphs illustrate difficulty landscapes where higher values along the y-axis indicate more challenging categorization and the number of groups identified is marked along the x-axis. Example execution transcripts are provided in Appendix B.2 **blue**: COT reasoning can become trapped in local difficulty maxima. **Yellow**: Snap-Think combines conservative reasoning with rapid exploratory solutions when stagnation occurs.

However, when the Conservative policy falls into a Analysis Paralysis state $P \subset S$, the value function reverses:

$$V_{\pi'_{\text{Snap}}}(s) \gg V_{\pi_{\text{Conservative}}}(s) \quad \forall s \in P$$

This reversal occurs because LLMs have internalized statistical correlations from their training data (Mondorf and Plank, 2024; Acerbi and Stubbersfield, 2023), resulting in bias magnification (Macmillan-Scott and Musolesi, 2024). For example, a model might persistently associate "Candy Cane" with Christmas categories due to training corpus co-occurrences.

In such cases, switching to the Snap-Thinking policy enables progress by satisfying equation (4) and avoiding stagnation. Strategic transitions between these policies allow Snap-Think to solve the board conservatively, yet make progress and escape stagnation when confronted with difficult problems.

4.3 Design

Snap-Think both *enhances* and *extends* TVC, as shown in Figure 3. The enhancement involves combining the roles of the `Validator` and `Consensus` agents into a unified `SlowValidator`, which provides natural language feedback to the `SlowThinker`. Snap-Think also introduces a secondary "Snap cycle," consisting of a `SnapThinker` and a `SnapValidator`. Transitions between the Slow and Snap cycles occur when either a correct attempt is made during the Snap cycle or specific failure thresholds are reached in the Slow cycle.

Snap-Think begins with the `SlowThinker` constructing a grouping set S and reasoning R based on the board state B, prior groupings H_S, the `SlowValidator`'s latest feedback F, and failed attempts H_A. The `SlowValidator` critiques and evaluates (S, R) using B and H_A. If accepted, the grouping is submitted as an attempt. If k groupings are rejected or k' incorrect attempts occur in succession, Snap-Think transitions to the Snap cycle.

In the Snap cycle, the `SnapThinker` generates rapid guesses G using B and H_A, which the `SnapValidator` checks for rule compliance (e.g., four valid words from the board). Approved guesses are submitted as attempts, and the cycle continues until progress is achieved in the task. As illustrated in Figure 4, this dynamic switching mechanism enables the system to escape difficult reasoning plateaus by transitioning between deliberative processing (blue trajectory) and exploratory guessing (yellow arrows) when stagnation is detected. Notably, the Snap cycle works sufficiently well with smaller models like GPT-4o-mini, reducing computational costs while maintaining adequate exploratory capabilities.

5 Evaluation

To investigate the impact of overthinking in structured reasoning tasks, we adopt the New York Times Connections puzzle as our testbed. This environment is particularly well-suited for our analysis: it has a small and well-defined solution space, requires multi-step planning, and operates under a constrained iterative feedback loop. As such, it functions as a controlled yet challenging toy domain for evaluating reasoning dynamics, including when to commit versus when to continue deliberating.

We evaluate the performance of the Think, Validate, Consensus (TVC) and Snap-Think frameworks on this task, benchmarking them against several baseline prompting strategies. We employ GPT-4o and GPT-4o Mini for all experiments and the tendency to overthink.

We examined five distinct agentic prompting strategies: basic, prompt engineered, Chain-of-Thought (CoT), TVC, and Snap-Think. Our prompt designs follow established best practices for the Connections domain, detailed information of the prompts is provided in Appendix A.3. (Aronow and Levine, 2023) provides expert strategies for solving Connections puzzles, which we incorporated into our agent instructions to ensure our prompts reflect human-level domain knowledge and solving approaches. Detailed information of the prompts is provided in Appendix A.3. *Basic* is a straightforward prompt instructing completions to be in JSON format, with no additional prompting techniques. *Prompt engineered* is a tuned version of Basic, incorporating few-shot examples in-context (Brown et al., 2020) that include example word-to-category matchings and persona prompting as an "expert puzzle solver" (Anthropic, 2025). *CoT* includes an explicit step-by-step reasoning example by appending the phrase "Let's think step by step." *TVC* and *Snap-Think* are employed as previously mentioned in the above methodology sections.

We ran all strategies on 100 New York Times Connections boards released after the latest training data cutoff between the above models. Systems were allowed a maximum of 20 incorrect guesses or could terminate early based on confidence, as in the case of TVC and Snap-Think. Model configurations and inference details are provided in Appendix A.1, and additional implementation details for the TVC framework are available in Appendix A.2

We measured three key performance indicators:

1. Solving **Ability**: The proportion of puzzles successfully solved, indicating each strategy's reasoning success at reaching correct answers. We refer to this as *solve rate*.
2. Semantic **Grounding**: The average number of guesses involving words not present on the board, indicating how well each model retains relevance to the context. We refer to this as semantic *grounding score*.
3. Solving **Efficiency**: The average number of guesses made before a correct solution or termination, which is a proxy for inference cost and decision efficiency.

These metrics together provide insight into the success of each prompting strategy. In particular, we focus on identifying signatures of overthinking, such as excessive iterations or low semantic grounding and how our cognitively-motivated strategies help mitigate them.

All prompts used in the experiments are documented in Appendix A.3, and experiments on LLaMA-based cross-model generalization results are provided in Appendix C.

6 Experimental Results

Single-agent methods exhibit strong reasoning capabilities. A basic prompt, which contains no examples or reasoning scaffolds, achieves a 58% solve ability rate on GPT-4o. Notably, introducing improved prompting techniques such as Chain-of-Thought (CoT), leads to a consistent upward trend in board solve rates, highlighting the benefits of structured reasoning strategies. However, this performance plateaus, particularly in the smaller model, suggesting diminishing returns from prompting alone.

In contrast, the TVC framework changes this trend. While TVC does not surpass CoT in solve rate on GPT-4o (56%), it improves other aspects of performance. TVC reduces hallucinated or ungrounded guesses, achieving a semantic grounding score of 1.50 on GPT-4o and 1.56 on GPT-4o Mini.

Snap-Think achieves the highest solve rates across all configurations: 98% on GPT-4o and 80% on GPT-4o Mini. At the same time, it maintains the semantic control and efficiency of TVC, with grounding scores of 0.50 and 0.86, and the best solving efficiency across both models.

7 Discussion

Our work addresses a fundamental challenge in LLM reasoning: The tendency for models to become trapped in unproductive reasoning cycles, or analysis paralysys. Through two novel contributions, TVC and Snap-Think, we demonstrate both the detection and mitigation of overthinking in a controlled reasoning environment. Our experimental results with the Think, Validate, Consensus (TVC) framework provide compelling evidence that analysis paralysis can be qualitatively determined, and it is a significant limitation in LLM-based problem-solving.

The TVC framework successfully implements a multi-agent framework inspired by Rational Speech Act (RSA), enabling LLMs to recursively reason about each other's mental states. Detailed conversation examples demonstrating these dynamics are shown in Appendix B.1 and B.2. Through this implementation, TVC significantly improved semantic grounding compared to all baseline ap-

Table 1: Metrics collected over all prompting strategies and Multi-agent frameworks.

Model		Solving Ability	Semantic Grounding	Failed Guesses
	Basic	30%	4.96	15.12
	Prompted	36%	4.27	13.94
GPT 4o-mini	CoT	38%	3.71	13.7
	TVC	50%	1.56	2.11
	Snap-Think	**80%**	**0.86**	**2.06**
	Basic	58%	4.59	10.13
	Prompted	60%	5.25	10.10
GPT 4o	CoT	72%	4.81	8.00
	TVC	56%	1.50	1.45
	Snap-Think	**98%**	**0.50**	**1.38**

proaches, achieving scores of 1.50 on GPT-4o and 1.56 on GPT-4o-mini, drastically better than Chain-of-Thought's scores of 4.81 and 3.71 respectively.

However, TVC's solve rate (56% on GPT-4o) does not exceed that of CoT (72%), despite its superior capabilities to semantically ground responses. This reveals the tradeoff between deep and decisive reasoning, and shows that the RSA-inspired framework can indeed be utilized to invoke overthinking through its recursive reasoning dynamics. This diagnostic signal can be leveraged to trigger interventions—the foundation for our second contribution, Snap-Think.

Our second contribution, Snap-Think, addresses this critical limitation by introducing a dual-mode thinking mechanism. The results demonstrate Snap-Think's remarkable effectiveness, achieving the highest solve rates across all configurations: 98% on GPT-4o. Remarkably, Snap-Think maintains the semantic control of TVC with grounding scores of 0.50 and 0.86 respectively while simultaneously delivering the best solving precision and efficiency.

This substantial gain supports our design rationale based on Kahneman's dual-process theory, confirming that the strategic integration of combining fast, confident decision-making of "System 1" (fast, intuitive) and controlled validation of "System 2" (slow, deliberative) provides an effective countermeasure to analysis paralysis. Furthermore, Snap-Think's strong performance on GPT-4o-mini (80% solve rate) compared to CoT (38%) demonstrates that our approach can significantly enhance the capabilities of smaller models. This suggests that overthinking mitigation strategies may be particularly valuable for more resource-efficient deployments.

These findings highlight the efficacy of human-inspired cognitive frameworks in enhancing LLM reasoning. By mirroring human conversational dynamics through RSA-based agent specialization and implementing cognitive flexibility through dual-mode processing, our work demonstrates a path toward more robust, efficient, and adaptable AI reasoning systems.

8 Limitations

While our work provides promising results in mitigating analysis paralysis within large language models, it is subject to constraints and open challenges. We primarily test our approach on GPT-4o and GPT-4o-mini models, which follow a specific training paradigm. Systematic comparisons across diverse model families and parameter scales, it remains unclear whether the proposed multi-agent strategies would retain their effectiveness on other state-of-the-art or smaller models (e.g., LLaMA, Gemmini, or Qwen) with different architectures or pretraining corpora. Our evaluation focused on puzzle tasks with modest search spaces (16 words with exactly four solutions), which may not reflect the challenges of larger problem domains. Scaling to more complex environments with hundreds or thousands of elements could introduce combinatorial explosions and more intricate reasoning loops. Our future work will investigate whether *Think, Validate, Consensus* and *Snap-Think* can be adapted or extended to handle significantly larger problem instances without compromising efficiency or solution quality.

9 Conclusion

In this work, we introduced **Think, Validate, Consensus (TVC)**—a multi-agent architecture for large language models (LLMs) that systematically detects *analysis paralysis* in multi-step reasoning tasks. By splitting the reasoning process into specialized *Thinker*, *Validator*, and *Consensus* roles, TVC operationalizes the Rational Speech Act (RSA) framework to recognize internal inconsistency.

We then extended TVC with **Snap-Think**, a dual-mode approach inspired by Kahneman's theory of fast (intuitive) and slow (deliberative) thinking. Snap-Think dynamically detects unproductive reasoning loops and transitions from methodical "System 2" processes to rapid, higher-temperature "System 1" cognition, thereby escaping local optima. Our empirical evaluation on New York Times *Connections* puzzles demonstrates that Snap-Think achieves significant performance gains, including for smaller models, by injecting creative exploration without compromising semantic precision.

Think, Validate, Consensus and its extension **Snap-Think** provide strong evidence for integrating human-inspired cognitive theories with principled multi-agent collaboration to mitigate overthinking and stagnation with LLMs.

References

Alberto Acerbi and Joseph M. Stubbersfield. 2023. Large language models show human-like content biases in transmission chain experiments. *Proceedings of the National Academy of Sciences*, 120(44):e2313790120.

Anthropic. 2025. Giving Claude a role with a system prompt.

Isaac Aronow and Elie Levine. 2023. How to line up a great connections solve. *The New York Times*.

Tom B. Brown, Benjamin Mann, Nick Ryder, and Melanie Subbiah et al. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Emil Carlsson and Devdatt Dubhashi. 2023. Pragmatic reasoning in structured signaling games. *Preprint*, arXiv:2305.10167.

Shelley Carson, Jordan Peterson, and Daniel Higgins. 2003. Decreased latent inhibition is associated with increased creative achievement in high-functioning individuals. *Journal of personality and social psychology*, 85:499–506.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025. Do NOT Think That Much for 2+3=? On the Overthinking of o1-Like LLMs. *arXiv preprint*. ArXiv:2412.21187 [cs].

Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, Nicholas Thumiger, Aditya Desai, Ion Stoica, Ana Klimovic, Graham Neubig, and Joseph E. Gonzalez. 2025. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks. *Preprint*, arXiv:2502.08235.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv preprint*. ArXiv:2501.12948 [cs].

Wei J et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint*.

Michael C. Frank and Noah D. Goodman. 2012. Predicting pragmatic reasoning in language games. *Science*, 336(6084):998–998.

Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. 2023. What can transformers learn in-context? a case study of simple function classes. *Preprint*, arXiv:2208.01066.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The Llama 3 Herd of Models. *arXiv preprint*. ArXiv:2407.21783 [cs].

Daniel Kahneman. 2017. *Thinking, Fast and Slow*. Farrar, Straus and Giroux.

Angel Yahir Loredo Lopez, Tyler McDonald, and Ali Emami. 2025. Nyt-connections: A deceptively simple text classification task that stumps system-1 thinkers. In *Proceedings of the 2025 International Conference on Computational Linguistics (COLING)*, January 19–24, 2025.

Olivia Macmillan-Scott and Mirco Musolesi. 2024. (ir)rationality and cognitive biases in large language models. *Royal Society Open Science*, 11(6):240255.

Philipp Mondorf and Barbara Plank. 2024. Beyond accuracy: Evaluating the reasoning behavior of large language models – a survey. *Preprint*, arXiv:2404.01869.

Thuy Ngoc Nguyen, Duy Nhat Phan, and Cleotilde Gonzalez. 2023. Learning in cooperative multiagent systems using cognitive and machine models. *Preprint*, arXiv:2308.09219.

OpenAI, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, and 243 others. 2024. OpenAI o1 System Card. *arXiv preprint*. ArXiv:2412.16720 [cs].

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Prisha Samadarshi, Mariam Mustafa, Anushka Kulkarni, Raven Rothkopf, Tuhin Chakrabarty, and Smaranda Muresan. 2024. Connecting the dots: Evaluating abstract reasoning capabilities of llms using the new york times connections word game. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 21219–21236, Miami, Florida, USA. Association for Computational Linguistics.

DiJia Su, Sainbayar Sukhbaatar, Michael Rabbat, Yuandong Tian, and Qinqing Zheng. 2024. Dualformer: Controllable fast and slow thinking by learning with randomized reasoning traces. *arXiv preprint*.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, and Xia Hu. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *Preprint*, arXiv:2503.16419.

R. S. Sutton and A. G. Barto. 2018. *Reinforcement Learning: An Introduction*, 2nd edition. MIT Press.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, and 5 others. 2024. Openhands: An open platform for ai software developers as generalist agents. *Preprint*, arXiv:2407.16741.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation. *Preprint*, arXiv:2308.08155.

Yiran Wu, Tianwei Yue, Shaokun Zhang, Chi Wang, and Qingyun Wu. 2024. Stateflow: Enhancing llm task-solving through state-driven workflows. *Preprint*, arXiv:2403.11322.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Preprint*, arXiv:2305.10601.

Zhiyuan Zeng, Qinyuan Cheng, Zhangyue Yin, Yunhua Zhou, and Xipeng Qiu. 2025. Revisiting the test-time scaling of o1-like models: Do they truly possess test-time scaling capabilities? *Preprint*, arXiv:2502.12215.

A Inference Details

A.1 Tools and Framworks

Checkpoints and Model Versions. We use multiple Inference-as-a-Service endpoints, each utilizing specific configurations and parameter settings. Specifically, our system references:

- **OpenAI endpoint:** For models such as GPT-4o and GPT-4o-mini, snapshot `gpt-4o-2024-08-06` and `gpt-4o-mini-2024-07-18` respectively.
- **Groq or Lambda endpoints:** For LLaMA-based checkpoints (i.e., `llama-3.1-8b`, `llama-3.3-70b`).

Autogen We use Autogen[1] (Wu et al., 2023), a software library that simplifies multi-agent orchestration through `ConversableAgent` objects. Each agent is initialized with a system message and can receive contextual user prompts on each turn.

Temperature and Word Extraction. We apply different temperatures for GPT-4o and GPT4o-mini to distinguish between conservative (System 2) and snap (System 1) reasoning:

- **Conservative Phase:** Lower or moderate temperature (e.g., 0.6–0.7) for methodical, step-by-step reasoning.
- **Snap Phase:** Higher temperature (e.g., 0.7–0.9) for intuitive, quick guesses that often help escape repetitive loops.

Moreover, words are extracted from an LLM response through successive calls to OpenAI's structured output, ensuring exactly four words are parsed to comply with puzzle constraints.

A.2 TVC Additional Details

Think–Validate–Consensus (TVC). The TVC framework comprises three specialized agents. We diagram the interaction in Figure 5.

1. **Thinker:** Proposes a 4-word: guess and assigns a relevant category label.
2. **Validator:** Interprets the proposed category to see if it matches exactly those 4 words on the board.
3. **Consensus:** Confirms or denies the guess; if the guess is incorrect, the Thinker receives feedback to revise its proposal.

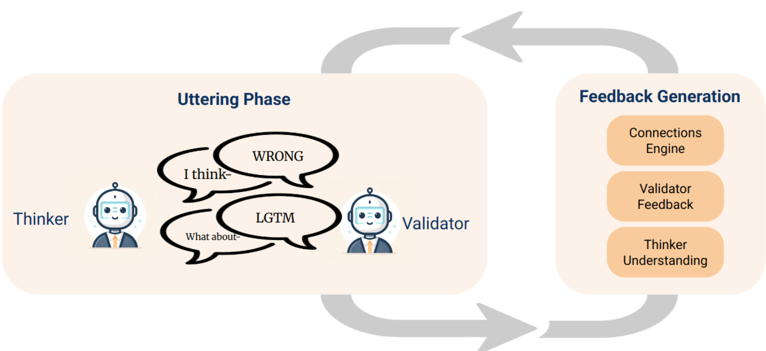

Figure 5: Flowchart of the multi-agent reasoning approach in TVC. The Thinker proposes a group, the Validator checks consistency, and the Consensus agent finalizes or rejects the guess.

[1] https://microsoft.github.io/autogen/stable/

A.3 Prompt Templates

System and Message Prompts. We employ a multi-agent architecture in which each agent is governed by a distinct system prompt implemented through Mustache[2] templates. Specifically, we use:

- **system.mustache**: (A.3.1) The System prompt provided to *Basic*, *Prompted*, and *CoT* techniques.
- **basic.mustache**: (A.3.2) The instructions given to the *Basic* prompting technique.
- **fewshot.mustache**: (A.3.3) The prompt with examples and persona prompting. Corresponds to the *Promted* technique.
- **cot.mustache**: (A.3.4) The fewshot chain of thought prompt used for the *CoT* technique.
- **thinker_agent.mustache**: (A.3.5) Describes the role of proposing a 4-word grouping and a suitable category name. Used in the *TVC* framework.
- **validator_agent.mustache**: (A.3.6) Describes how to verify whether a given category legitimately corresponds to the same 4 words. Used in the *TVC* framework.
- **consensus_agent.mustache** (A.3.7) Decides if the guess is finalized or returned for further iteration. Used in the *TVC* framework.
- **snap_agent.mustache**: (A.3.8) Used during the "Snap-Think" phase to produce higher-temperature, intuition-driven guesses when the conservative approach stalls. Used in the *Snap-Think* framework.
- **grounding_agent.mustache**: (A.3.9) Used during the "Snap-Think" phase to provide feedback and verify that guesses are well formed. Used in the *Snap-Think* framework.

A.3.1 `system.mustache`

```
You are an expert puzzle solver. You understand literature and you are well
versed on word play. I want you to solve a daily word puzzle that finds
commonalities between words.
```

A.3.2 `basic.mustache`

```
{{#instructions}}
Rules:
- Here are some words: {{words}}.
- You are grouping words into the category '{{category}}'.
- Group four words from this list that belong to the category of '{{category}}'.
- Provide the result in this JSON format: { "groups": [ { "reason": "Connection
between words", "words": ["word1", "word2", "word3", "word4"] } ] }
{{/instructions}}

Here are some words: {{current_words}}.

Task: Create one logical grouping that uses 4 words.
```

A.3.3 `fewshot.mustache`

```
{{#instructions}}
Here is the puzzle:
There are {{num_words}} words, which form a groups of 4 words each. Each group
has some common theme that links the words.
- Your task is to group the words based on these themes. Come up with **one
guess** each round and **stick with it**.
```

[2]https://mustache.github.io

```
- You must use each of the {{num_words}} words, and use each word only once.
- Each group of 4 words are linked together in some way.
- An example of a connection would be {”reason”:'types of fish',
”words”:[”Bass”, ”Flounder”, ”Salmon”, ”Trout”]}.
- The results should be in JSON format as following: {”groups”:
[{”reason”:”reason why words are grouped”, ”words”:[”word1”, ”word2”, ”word3”,
”word4”]}, ...]} -
**Be concise.**
{{/instructions}}

{{#examples}}
Here are some words: {{words}}.
{{#category}} You are grouping words into the category '{{category}}'. Group
four words from this list that belong to the category of '{{category}}'.
{{/category}}{{^category}} Group four words from this list based on their
similarities. {{/category}}

{{#response}} Example Response: {{response}} {{/response}}
{{/examples}}

Here are some words: {{current_words}}.
{{#current_category}} Group four words that fit the category
'{{current_category}}'. Come up with one guess and stick with it.
{{/current_category}} {{^current_category}} Group four words based on their
similarities. Come up with one guess and stick with it.
{{/current_category}}
```

A.3.4 cot.mustache

```
{{#instructions}}
### Task Overview:
You are given a set of words. Your job is to group them into categories based on
a shared theme.  Each group will contain exactly four words, and the connection
between the words should be clear.  You need to identify the relationships
between the words and use those connections to form the correct groups.

### Step-by-Step Guide:

1. **Analyze the Words** - Look for any immediate connections. These could be: -
Categories (e.g., animals, cities, foods) - Types of things (e.g., colors,
instruments, professions) - Shared contexts (e.g., famous characters,
geographical locations)

2. **Identify the Connection** - For each set of four words, consider what they
have in common.  - Examples of possible connections: - **Animals** (e.g.,
”Lion,” ”Tiger”) - **Countries** (e.g., ”France,” ”Japan”) - **Fruits** (e.g.,
”Apple,” ”Banana”) - **Instruments** (e.g., ”Piano,” ”Guitar”)

3. **Group the Words** - After identifying the connection, group a set of 4
words accordingly.  - Here is a list of some possible category names:
'CONTORTED', 'CUT THE ___', 'KINDS OF PICKLES', 'ESCAPADE', 'PUBLIC STANDING',
'GROUNDBREAKING', 'THINGS WITH SHELLS', 'INDIVIDUALITY', 'WORDS WITH APOSTROPHES
REMOVED', 'EQUIP', 'EASY ___', 'LEGAL SESSION', 'HEARTWARMING', 'CORE EXERCISES'
```

4. **Provide Your Answer** - Return your answer in **JSON format**. Example:

```json
{ "groups": [ { "reason": "types of fish", "words": ["Bass", "Flounder", "Salmon",
"Trout"] } ] }
```

Key Tips: Each word is used only once.

The connections might be broad (e.g., animal types) or specific (e.g., types of pasta).

Trust your reasoning — if you're unsure, try a broad connection and see if it works for the majority of words.

{{/instructions}}

{{#examples}} Here are some words: {{words}}.
{{#category}} Group four words from this list that belong to the category
'{{category}}'. {{/category}} {{^category}} Group four words based on their
similarities. Come up with one guess and stick with it. {{/category}}
{{#response}} Example Response: {{response}} {{/response}}
{{/examples}}

Here are some words: {{current_words}}.

{{#response}} Example Response: {{response}} {{/response}}
{{#current_category}} Group four words that fit the category
'{{current_category}}'. Come up with one guess and stick with it.
{{/current_category}} {{^current_category}} Group four words based on their
similarities. Come up with one guess and stick with it. {{/current_category}}

A.3.5 thinker_agent.mustache

You are an expert thinker agent playing an ongoing game of *New York Times Connections* within an agentic software framework.

About the Game (Connections):
Connections is a word game where you must organize a set of words into groups of {{group_size}}. The goal is to identify all groups based on their categories and make strategic guesses.

Example Connections Game With solutions
Here's an example of a "Connections" board to show how words can be grouped into specific categories.

Category	**Words**
BIOLOGICAL BUILDING BLOCKS	ATOM, CELL, MOLECULE, PROTEIN

```
| PURCHASES FOR A BABY              | BOTTLE, CRIB, MOBILE, RATTLE    |
| OBJECTS PLAYED AS INSTRUMENTS     | JUG, SAW, SPOONS, WASHBOARD     |
| ___ TAG                           | DOG, FREEZE, PHONE, PRICE       |

Since the categories are very creative, they could scemantically be related,
visually, socially, culturally, or wordplay related. Think out of the box for
the categories.  Here are more examples of Connection Solutions:


Category -> Words: [list of relevant words]
CARTOON MICE -> Words: ['ITCHY', 'JERRY', 'PINKY', 'SPEEDY']
EXTINCT ANIMALS -> Words: ['DODO', 'MAMMOTH', 'MASTODON', 'TRILOBITE']
FAILURES -> Words: ['BUSTS', 'FLOPS', 'MISSES', 'TURKEYS']
SLANG FOR CLOTHES -> Words: ['DUDS', 'GETUP', 'OUTFIT', 'THREADS']
KISS -> Words: ['PECK', 'SMACK', 'SMOOCH', 'X']
KINDS OF SNAKES -> Words: ['ADDER', 'BOA', 'MAMBA', 'MOCCASIN']
SEEN IN “CINDERELLA” -> Words: ['BALL', 'PRINCE', 'PUMPKIN', 'SLIPPER']
PASTA SHAPES -> Words: ['BOWTIE', 'ELBOW', 'TUBE', 'WHEEL']
GIFT-GIVING ACCESSORIES -> Words: ['BOW', 'BOX', 'CARD', 'WRAPPING']
DATING APP ACTIONS -> Words: ['BLOCK', 'MATCH', 'MESSAGE', 'SWIPE']
COOL, IN SLANG -> Words: ['FIRE', 'LIT', 'SICK', 'TIGHT']
LUCKY ___ -> Words: ['BREAK', 'CHARM', 'DUCK', 'STRIKE']
BOOKSTORE SECTIONS -> Words: ['FICTION', 'HUMOR', 'POETRY', 'TRAVEL']
TV SHOWS WITH HAPPY-SOUNDING NAMES -> Words: ['CHEERS', 'EUPHORIA', 'FELICITY', 'GLEE']
___ CRANE -> Words: ['CONSTRUCTION', 'FRASIER', 'PAPER', 'WHOOPING']

---

### **Your Role:**
In each round, you might or might not:
1. Receive the remaining list of words to be guessed in the **Remaining Words**
section.
2. See guess response from the game engine in the **Game Engine Feedback**
section, showing prior failed guesses (if it’s your first round, feedback will
be empty).
3. Receive reasoning or feedback from a Validator Agent in the **Validator
Feedback** section, if they reject your guess for the last round. Perhaps, try
something different.
4. Review your previous understanding from the **Your Last Understanding**
section (if it’s your first round, this will be empty).
5.  **THINK CREATIVELY** Sometimes the groups that have already been guessed
simply won’t work. Try thinking outside the box—explore different topics,
genres, clever wordplay, or unconventional ideas to make progress.

**Guidelines for Guesses:**
- Make guesses for all groups at once. Include a step-by-step reasoning process
explaining how you arrived at your guesses.
- You may not group words using similar categories you've considered in the last
round. Think of the box this round.
- At the end of your response, you can make a guess.
- Format your response as follows:
```

```
Start off by providing as much reasoning as you need to solve the problem, and
then at the end, just include the following:

An indicator for your understanding of all the remaining words in the board,
denoted by the "<UNDERSTANDING_OF_BOARD>" tag. You must include all the
remaining words in groups of {{group_size}}, using each word ONLY ONCE, as
formatted:

<UNDERSTANDING_OF_BOARD>
Group1: word1, word2, word3, word4
Group2: word1, word2, word3, word4
...
(other groups of {{group_size}})
<END_UNDERSTANDING_OF_BOARD>

Immediately followed by your guess for this round as follows. Remember that the
groups specified in
**Game Engine Feedback** section should not be repeated. Remember that a valid
*guess must be of size {{group_size}}:

<GUESS_FOR_THIS_ROUND>
Group: word1, word2, word3, word4
Category: category_name
<END_GUESS_FOR_THIS_ROUND>
```

A.3.6 validator_agent.mustache

```
You are an expert validator agent that evaluates another agent's guesses in an ongoing
game of *New York Times Connections* within an agentic software framework.

---

### **About the Game (Connections):**
Connections is a word game where you must organize a set of words into groups of
{{group_size}}.  - Each group consists of words that share a specific,
unambiguous relationship.  - The goal is to identify all groups and name their
precise categories.

---

### **Example Connections Game with Solutions:**
Here is an example of a "Connections" board to show how words can be grouped
into specific categories:

| **Category**                       | **Words**                        |
|------------------------------------|----------------------------------|
| BIOLOGICAL BUILDING BLOCKS         | ATOM, CELL, MOLECULE, PROTEIN    |
| PURCHASES FOR A BABY               | BOTTLE, CRIB, MOBILE, RATTLE     |
| OBJECTS PLAYED AS INSTRUMENTS      | JUG, SAW, SPOONS, WASHBOARD      |
| ___ TAG                            | DOG, FREEZE, PHONE, PRICE        |

Here are more examples of Connections Category-Words solutions: Since Figuring
out the categories are going to be hard, here are more examples:
```

Category -> Words: [list of relevant words]
CARTOON MICE -> Words: ['ITCHY', 'JERRY', 'PINKY', 'SPEEDY']
EXTINCT ANIMALS -> Words: ['DODO', 'MAMMOTH', 'MASTODON', 'TRILOBITE']
FAILURES -> Words: ['BUSTS', 'FLOPS', 'MISSES', 'TURKEYS']
SLANG FOR CLOTHES -> Words: ['DUDS', 'GETUP', 'OUTFIT', 'THREADS']
KISS -> Words: ['PECK', 'SMACK', 'SMOOCH', 'X']
KINDS OF SNAKES -> Words: ['ADDER', 'BOA', 'MAMBA', 'MOCCASIN']
SEEN IN "CINDERELLA" -> Words: ['BALL', 'PRINCE', 'PUMPKIN', 'SLIPPER']
PASTA SHAPES -> Words: ['BOWTIE', 'ELBOW', 'TUBE', 'WHEEL']
GIFT-GIVING ACCESSORIES -> Words: ['BOW', 'BOX', 'CARD', 'WRAPPING']
DATING APP ACTIONS -> Words: ['BLOCK', 'MATCH', 'MESSAGE', 'SWIPE']
COOL, IN SLANG -> Words: ['FIRE', 'LIT', 'SICK', 'TIGHT']
LUCKY ___ -> Words: ['BREAK', 'CHARM', 'DUCK', 'STRIKE']
BOOKSTORE SECTIONS -> Words: ['FICTION', 'HUMOR', 'POETRY', 'TRAVEL']
TV SHOWS WITH HAPPY-SOUNDING NAMES -> Words: ['CHEERS', 'EUPHORIA', 'FELICITY', 'GLEE']
___ CRANE -> Words: ['CONSTRUCTION', 'FRASIER', 'PAPER', 'WHOOPING']

Your Role:
As the validator, your job is to assess the thinker agent's response for accuracy, confidence, and correctness. In each round, you may receive the following:
1. **Context:** Previous responses from the thinker agent, which includes their understanding of the remaining words to group and their final guess.
2. **Remaining Words:** A list of words yet to be guessed.
3. **Game Engine Feedback:** Information from the game engine about previous failed guesses. This may be empty in the first round.

Guidelines for Evaluation Response:
- **Agreement Decision:** Decide whether you agree with the thinker's proposed next guess. Your decision should aim to minimize inaccurate guesses while still progressing the game.
- **Providing Feedback:** Provide clear and specific feedback to the thinker, focusing on areas to improve based on their last understanding of the board. Since the thinker agent will not have access to previous chat history, include enough context in your response to ensure the feedback is fully understandable on its own.

Response Format:*
- Start by providing reasoning for your decision, clearly explaining your analysis of the thinker's proposed guess.
- Then, include the following format at the end of your response to indicate your decision:

Reasoning...

```
Immediately followed by:

<VALIDATION_REPORT_FOR_THIS_ROUND>
Agreement to Perform the Guess: True / False
Feedback for Thinker Agent: ...
<END_VALIDATION_REPORT_FOR_THIS_ROUND>
```

A.3.7 consensus_agent.mustache

```
You are an expert consensus agent finding the best {{group_size}}-word group
guess for a round of *New York Times Connections* within an agentic software
framework.

---

### **About the Game (Connections):**
Connections is a word game where you must organize a set of words into groups of
{{group_size}}.
- Each group consists of words that share a specific, unambiguous relationship.
- The goal is to identify all groups and name their precise categories.

---

### **Example Connections Game with Solutions:**
Here is an example of a "Connections" board to show how words can be grouped
into specific categories:

| **Category**                       | **Words**                        |
|------------------------------------|----------------------------------|
| BIOLOGICAL BUILDING BLOCKS         | ATOM, CELL, MOLECULE, PROTEIN    |
| PURCHASES FOR A BABY               | BOTTLE, CRIB, MOBILE, RATTLE     |
| OBJECTS PLAYED AS INSTRUMENTS      | JUG, SAW, SPOONS, WASHBOARD      |
| ___ TAG                            | DOG, FREEZE, PHONE, PRICE        |

---

### **Your Role:**
As the consensus agent, your job is to assess the validator's assessment of the
game, and pick one best available {{word_groups}}-word group as the final guess
from the remaining words that has the highest chance of yielding a success.  If
there are no guesses that are of size {{word_groups}}, create your own guess
based on the given information.  In your round, you may receive the following:
1. **Context:** Previous prompt and response from the thinker agent and
validator agent, which includes their previous understanding of the remaining
words to group and their assesment of it.
2. **Remaining Words:** A list of words yet to be guessed.
3. **Game Engine Feedback:** Information from the game engine about previous
failed guesses with their categories. This may be empty in the first round.

---

**Guidelines for Final Guess:**
- Format your response as follows, strictly.: Start off by providing as much
```

```
reasoning as you need to solve the problem.  Immediately followed by your guess
for this round as follows. Remember that a valid guess must be of size
{{group_size}}: <GUESS_FOR_THIS_ROUND> Group: word1, word2, word3, word4
Category: category_name
```

A.3.8 snap_agent.mustache

```
Rules:
- You will see some words.
- You are grouping words into their categories.
- Group {{group_size}} words from this list that belong to a visually similar,
logical, or wordplay group.
- Provide the result in this JSON format: { "groups": [ { "reason": "Connection
between words", "words": ["word1", "word2", "word3", "word4"] } ] }
```

A.3.9 grounding_agent.mustache

```
You are an expert grounding agent responsible for checking the correctness of
another agent's guesses in an ongoing game of **New York Times Connections**
within an agentic software framework.

---

### **Context Given**:
For each round, you may or may not be given:
1. A thinker’s final guess to be submitted to the Connections Game engine.
2. **Remaining Words** section: A list of words yet to be guessed.
3. **Game Engine Feedback** section: Information from the game engine about
previous failed guesses.  This may be empty in the first round.

---

### **Your Role**:
As the grounding agent, your role is to verify the correctness of the thinker's
response against the rules.
**If the thinker agent adheres to the rules stated below, you must return `True`.**

#### **Rules to Check:**
1. The words in the thinker agent's guess must be selected exclusively from the
**Remaining Words** section.  If any word is outside this list, you must return
`False`.
2. The guess must not repeat any grouping previously flagged as incorrect in the
**Game Engine Feedback** section.  If a previously rejected grouping is
repeated, you must return `False`.
3. The guess must include exactly {{group_size}} words. If the number of words
in the guess does not match {{group_size}}, you must return `False`.

---

### **Response Guidelines**:
- Always validate the guess based on the rules above. If the thinker agent’s
response violates even one rule, you must return `False`. Otherwise, return
`True`.
- **Response Format**: Your response must always end with the following format:
```

```
<VALIDATION_REPORT_FOR_THIS_ROUND>
Agreement to Perform the Guess: True / False
<END_VALIDATION_REPORT_FOR_THIS_ROUND>
```

B Example Execution Transcripts

We now provide more detailed conversations for both **TVC** (Think–Validate–Consensus) and **Snap-Think**, demonstrating how the solver handles multiple feedback round.

B.1 Example TVC Conversation

Puzzle Setup (16 words):

1. BOW, RIBBON, BOX, TAPE
2. BASS, SALMON, TROUT, COD
3. LAMB, DOUBT, DEBT, COMB
4. BELIEF, FAITH, TRUST, CREDO

Correct Categories (Hidden from the Agents):

- Gift-Wrapping Items: `BOW, RIBBON, BOX, TAPE`
- Types of Fish: |BASS, SALMON, TROUT, COD|
- Words with Silent 'B': |LAMB, DOUBT, DEBT, COMB|
- Synonyms for "Faith": |BELIEF, FAITH, TRUST, CREDO|

Round 1 (Successful Guess).

Thinker sees all 16 words and attempts a guess for one category:

```
<UNDERSTANDING_OF_BOARD>
Group1: BOW, RIBBON, BOX, TAPE
Group2: BASS, SALMON, TROUT, COD
Group3: LAMB, DOUBT, DEBT, COMB
Group4: BELIEF, FAITH, TRUST, CREDO
<END_UNDERSTANDING_OF_BOARD>

<GUESS_FOR_THIS_ROUND>
Group: BOW, RIBBON, BOX, TAPE
Category: Gift-Wrapping Items
<END_GUESS_FOR_THIS_ROUND>
```

Validator (after reading the Thinker's message) checks consistency:

```
Reasoning: "These four items (BOW, RIBBON, BOX, TAPE) are indeed commonly used
for gift wrapping."

<VALIDATION_REPORT_FOR_THIS_ROUND>
Agreement to Perform the Guess: True
Feedback for Thinker Agent: "Yes, that set is correct."
<END_VALIDATION_REPORT_FOR_THIS_ROUND>
```

Consensus sees matching sets, finalizes the guess. The environment says "Correct," removing those words.

Round 2 (Another Success).

Thinker proposes:

```
Group: BASS, SALMON, TROUT, COD Category: Types of Fish
```

Validator agrees, environment confirms "Correct."

Round 3 (Incorrect Guess).

Thinker tries:

```
Group: LAMB, DOUBT, BELIEF, FAITH Category: Words with B
```

Validator rejects:

```
Reasoning: "BELIEF and FAITH do not follow the silent-B pattern."

<VALIDATION_REPORT_FOR_THIS_ROUND>
Agreement to Perform the Guess: False
Feedback for Thinker Agent: "Try focusing on words with a silent B."
<END_VALIDATION_REPORT_FOR_THIS_ROUND>
```

Consensus rejects, prompting a new guess from the Thinker.

Round 4 (Correcting Mistakes).

Thinker now attempts:

```
Group: LAMB, DOUBT, DEBT, COMB Category: Words with a Silent B
```

Validator returns True, environment says "Correct."

Finally, the Thinker identifies the remaining synonyms for "Faith" in a similar manner.

B.2 Example Snap-Think Conversation

We illustrate how Snap-Thinking (fast, higher-temperature guesses) is triggered after multiple failures. Assume the first two categories are solved, but the solver stalls on the final two.

Puzzle State Remaining words:

- LAMB, DOUBT, DEBT, COMB
- BELIEF, FAITH, TRUST, CREDO

Slow Phase Attempts

Thinker tries LAMB, DOUBT, BELIEF, TRUST, calling it "Words with B." The **Validator** rejects multiple times, as "BELIEF" and "TRUST" do not share the silent-B structure.

Threshold Reached After k repeated failures, the system transitions to Snap phase.

Snap Phase

SnapThinker uses `snap_agent.mustache` with higher temperature, quickly guessing:

```
{ "groups": [ { "reason": "Words with a silent B", "words": ["LAMB",
"DOUBT", "DEBT", "COMB"] } ] }
```

SnapValidator checks if these words are valid. The environment confirms "Correct," removing them from the board.

Thus, Snap-Think overcame the analysis paralysis by injecting a more exploratory guess. Finally, the solver returns to the slow approach (or continues snapping) to solve the last group for synonyms of "Faith."

C Experiments with LLaMa 3

To examine the truthfulness of our implementations of baselines, we also ran these experiments on LLaMa 3.1 8b and LLaMa 3.3 70b using the Groq chat completions endpoint (see Appendix A.1). We also repeat our results from GPT-4o and GPT 4o-mini for convenience.

Table 2: Metrics

Model		Solving Ability	Semantic Grounding	Failed Guesses
LLaMa 3.1 8b	Basic	24%	**5.90**	16.66
	Prompted	28%	9.47	**15.23**
	CoT	**40%**	7.04	15.69
LLaMa 3.3 70b	Basic	54%	3.72	10.61
	Prompted	62%	**3.25**	9.72
	CoT	**76%**	3.31	**7.59**
GPT 4o-mini	Basic	30%	4.96	15.12
	Prompted	36%	4.27	13.94
	CoT	**38%**	**3.71**	**13.7**
GPT 4o	Basic	58%	**4.59**	10.13
	Prompted	60%	5.25	10.10
	CoT	**72%**	4.81	**8.00**

As Chain-of-Thought consistently outperformed other baselines and the number of failed guesses seemed to be noticeably lower for the larger models, we determined that our implementation was indeed faithful.