COLLECTIVE INTELLIGENCE - DNA MODEL

Ye Yuhe (Jason) jason.ye@keio.jp
Gu Yu ygu.cju@gmail.com
Federico Peitti fpeitti@keio.jp
Yu Yueyang (Sakana) yakuyou.jp@gmail.com

ABSTRACT

This report explores the development of a Collective Intelligence DNA Model, an AI framework inspired by human cognition and collaborative intelligence. While large language models (LLMs) such as OpenAI's GPT-4 and Microsoft's LLaMA exhibit remarkable problem-solving capabilities, they are inherently limited as standalone entities. Human intelligence, by contrast, thrives through specialization, hierarchy, and structured decision-making. This research seeks to emulate these principles by introducing a hierarchically structured multi-agent system that improves AI collaboration and problem-solving efficiency.

At the core of the model is the DNA node, responsible for task allocation and iterative evaluation. Supporting it are stem cell nodes, which represent specialized agents collaborating to refine responses through discussion, and an aggregator node, which consolidates diverse insights. A key inspiration for this model is hierarchical organization systems like military rank structures, which efficiently distribute authority and decision-making responsibilities. To explore this concept, this research integrates an Army Hierarchy framework, a structured ranking-based system that mirrors the DNA Model's decision-making architecture. By mapping AI agents onto a military-style chain of command, this approach optimizes information flow, task delegation, and problem-solving efficiency.

1 Overview

Background and Overview

Artificial intelligence (AI) has made remarkable strides in recent years, with large language models (LLMs) such as OpenAI's GPT-4 achieving capabilities comparable to, and sometimes surpassing, human performance in various domains. These models have demonstrated proficiency in natural language processing (NLP), reasoning, and complex problem-solving tasks. However, while GPT-4

and similar LLMs exhibit remarkable individual performance, they remain limited as single-agent systems, often struggling with inconsistencies, contextual misunderstandings, and generating highly domain-specific insights.

Interestingly, humans, the dominant species on Earth, provide a compelling example of the power of collective intelligence. [1] Human societies excel not only due to individual abilities but because of their capacity to collaborate, leveraging diverse skills and perspectives to solve challenges. At the same time, each human being can be seen as a multi-agent system, where the brain's cortical regions—organized into specialized areas analogous to agents—coordinate seamlessly to produce unified, intelligent behavior. Studies suggest that cortical circuits exhibit a level of specialization and communication that rivals the capabilities of large AI systems, forming an integrated and adaptive decision-making unit. [1][2]

Building on this inspiration, this research aims to enhance the collective intelligence of AI systems by designing multi-agent frameworks that emulate human-like collaborative structures. Specifically, we develop a framework for assessing and improving the collective capabilities of multi-agent systems. Unlike approaches that modify the internal mechanisms of individual agents, our method optimizes the social topology and communication protocols of the collective system using evolutionary algorithms. This allows the system to evolve and adapt, forming more effective collaborative structures without altering the pretrained agents themselves.

To evaluate the system, we will introduce a performance metric based on spectral analysis of standard benchmark scores. This metric provides a generalizable way to assess the emergent intelligence of the collective system, offering insights into its capabilities across diverse tasks.

In this research, we implemented and compared two prominent multi-agent frameworks:

1. **OpenAI Swarm Framework with GPT-4:** A foundational platform for orchestrating agent interactions and role allocation. [3]

2. **Microsoft Autogen Framework with LLaMA 3.2 (3B):** A more flexible and scalable system integrated with Ollama, offering advanced features like local processing without the need of using API. [4] [5]

By combining the strengths of swarm intelligence and multi-agent collaboration, this work seeks to address the intrinsic limitations of standalone LLMs and move toward robust, decentralized AI systems. Just as humans as a species have leveraged their collective intelligence to overcome complex challenges, this research aims to unlock the potential of AI as a collective system, paving the way for scalable, adaptive, and contextually aware AI frameworks.

2 Model

2.1 Introduction

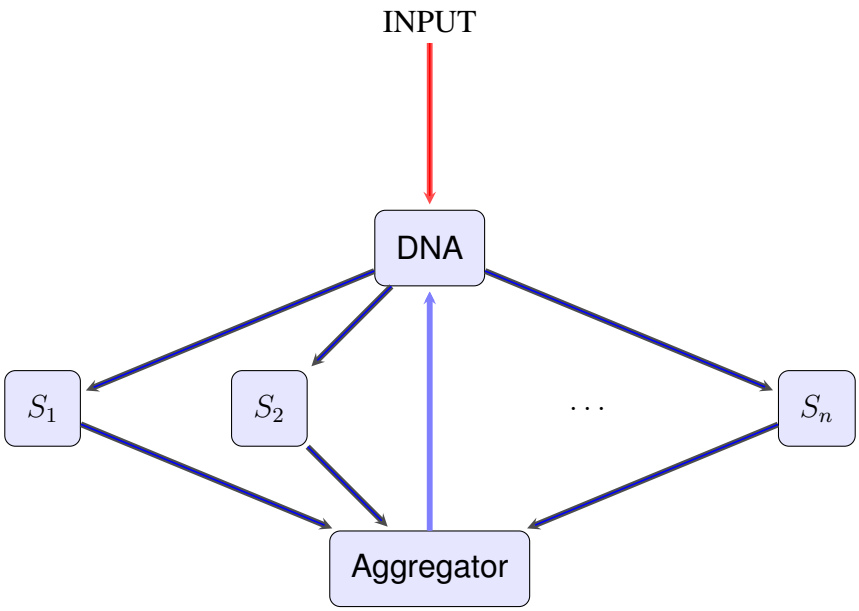

Figure 1: Graph depicting the basic idea of the model.

The model consists of a cyclic feedback loop that can be simplified as follows. The DNA node receives a problem to solve as input and decides how many specialist nodes (referred to as stem cell nodes) and how many agents will be needed to solve that problem. Consequently, the specialist groups answer and debate within the group for an indefinite number of rounds. Once the debate rounds are over, every agent passes their answer to the aggregator node, who will be in charge of summarizing the responses from each of the n stem cell nodes while still maintaining the contents. Finally, the aggregator node will pass this summary back to the DNA node, who will evaluate the responses and choose whether the final answer is satisfactory or not. If it is, then the final answer will be provided by the DNA node from the information gathered. If not, the DNA node will decide whether a rearrangement of agents or specializations is needed and have another iteration of the model until it is satisfied with the final answer. In figure cfig:basicmodel, a basic idea of the model can be observed. Each S_i on the graph will be a stem cell node.

2.2 DNA node

The DNA node is the central and most critical component of this structure. It specifically plays two important distinctive roles:

1. **Determining Specialist Groups:** When a query is received, the DNA node analyzes its complexity and requirements. Based on this analysis, it determines the optimal number of specialists needed and assigns specific roles to each. These roles are tailored to the context of the query, such as economists, politicians, or scientists, depending on the domain of the question and are freely chosen.

2. **Evaluating Specialist Outputs:** Once the specialist groups have collectively worked on the query, their results are aggregated and sent back to the DNA node. At this stage, the DNA node critically evaluates the quality of the aggregated answers and decides the next step, which, as explained previously, depends if the DNA believes if consensus has been reached and the provided answer is satisfactory (makes sense given the context and is supported by explanations).

Over time, the DNA node will improve the specialization assignments based on feedback from previous interactions. By analyzing past performance and results, it can make more effective decisions about group composition, role assignment, and evaluation criteria. This self-adaptive capability ensures that the system becomes increasingly capable of addressing questions with higher precision.

2.3 Stem Cell node

The n stem cell nodes, activated by the DNA node, play a key role in the generation of diverse and comprehensive answers to a given question. Each stem cell node operates within a designated role (e.g. economist, politician, scientist) and contributes to solving the query using specific methods and perspectives based on that particular role. The process of activating multiple stem cell nodes with different roles to answer the same question is similar to initiating weights of different distributions during deep learning. Each stem cell node approaches the question independently, making full use of its specific role to approximate the true answer from various points of view. This diversity not only can enhance the structure's ability to detect flaws by identifying significant inconsistencies between answers, but also improves comprehensiveness by integrating multiple methodologies and viewpoints into the problem-solving process.

After generating individual responses, the specialists assigned to the same role are grouped together for a simulated discussion. Within each discussion, the specialists will review its initial answer and compare it with the responses of others for an indefinite number of rounds. Each agent then updates its response based on insights gained from others in the same stem cell node, which fosters collaborative refinement, allowing the group to collectively approximate a more accurate and robust response.

After updating their responses, each agent makes one of two decisions:

1. **Satisfactory**: If the node is confident in its updated answer, it indicates satisfaction and finalizes its response.
2. **Dissatisfactory**: If the node finds its updated answer insufficient or considers further improvement is possible, it requests another round of discussion. In this case, the node will incorporate the refined answers of others into its own response during the next iteration.

This iterative cycle continues until at least half of the agents in the stem cell node are satisfied.

Once the within-group discussion concludes, all the individual responses from agents in the stem cell node are aggregated into a single combined response. This aggregated answer represents a consensus or best approximation from the group and is then passed to the DNA node for evaluation.

We believe the stem cell nodes have the following advantages over single agent models:

1. **Improved Accuracy**: having rounds of discussion and diverse perspectives allow errors and inconsistencies to be identified and resolved.
2. **Dynamic Collaboration**: The mechanism of simulated discussions ensures that each agent contributes meaningfully while adapting to the contributions of others.
3. **Scalability:** This structure can accommodate a wide range of roles, question subjects and difficulty levels, making it flexible for various types of queries.

By building a collaborative and iterative process among specialists, stem cell nodes should ensure that the aggregated response is not only well-rounded but also robust against potential flaws or oversights.

2.4 Aggregator node

The aggregator node serves as an intermediary between stem cell nodes and the DNA node, playing a crucial role in organizing and summarizing information while reducing computational overhead for the DNA node to process. The primary role of the aggregator node is to combine the responses of all specialists within a group into a single consolidated answer. This ensures that the DNA node receives a streamlined input, thereby reducing its computational burden and allowing it to focus on higher-level decision-making. The aggregator node **DOES NOT** assess the quality of individual responses or assign different weights to them. Instead, it prioritizes maintaining the originality of each contribution, which ensures that every specialist's perspective is represented, and prevents the loss of potentially critical insights. While aiming to preserve the depth and diversity of the input, the aggregator node also considers efficiency. It combines the answers in a manner that balances the richness of the content with the need to avoid redundancy, unnecessary verbosity or repetition.

3 Documentation on Army Hierarchy

3.1 Overview

This version of the code constructs a hierarchical graph based on a custom ranking system, in this case, the U.S. Air Force rank hierarchy. Each node in the graph corresponds to a rank, represented not by its name but by a symbolic representation (e.g., stars for generals, different symbols for enlisted ranks). The logic is similar to previous versions that used letters:

- Each character in the input string (A, B, C, ..., up to V) maps to a specific rank level.
- The hierarchy is top-down, with A representing the highest rank (General of the Air Force) and subsequent letters representing progressively lower ranks.
- Sibling nodes (nodes of the same level under the same parent) are connected horizontally to show their relationship.
- The # character breaks the sibling chain, ensuring that a node following # will not link to its previous sibling.

3.2 Key Concepts

1. **Hierarchy by Rank:**
   - The code uses a predefined list of U.S. Air Force ranks.
   - Each letter (A through V) corresponds to one of the 22 ranks, from highest to lowest.
2. **Parent-Child Relationships:**
   - A node representing a particular rank connects to the most recently created node of the immediately higher rank.
   - For example, if a node corresponds to 'Lieutenant General (O-9)' (C level), it connects to the most recently placed 'General (O-10)' (B level) node.
3. **Sibling Connections:**
   - When multiple nodes share the same parent and appear consecutively, they link to each other as siblings.
   - This sibling link is a horizontal connection in the hierarchy, providing visual continuity and clearly grouping children of the same parent.
4. **Breaking Sibling Chains with #:**
   - The # character acts as a reset switch for sibling connections.
   - After encountering #, the next rank node will connect to its parent but not to its previous sibling, effectively starting a new sibling chain from that point onward.

5. **Symbolic Representation:**
    - Instead of displaying the full rank names, each node is labeled with a symbolic representation.
    - For example, a 5-star rank might be shown as "★★★★★", while enlisted ranks are shown using different symbols.
    - This allows for a more compact and visually distinctive representation of the hierarchy.

3.3 Data Structures

- `rank_symbols`: A list of symbols corresponding to each rank level. The index in this list matches the rank level derived from the character.
- `last_node_of_letter`: Maps each rank level to the last created node of that level, allowing the code to determine the correct parent node for new nodes.
- `parent_last_child`: Maps a parent node to its most recently created child, enabling sibling links between consecutive children of the same parent.

3.4 Key Concepts

1. **Input String Parsing:**
    - The input string is examined character by character.
    - Letters (A to V) are converted to a rank level index and thus to a rank symbol.
    - Nodes are created and linked to their parents (if not top-level).
    - Sibling links are created between consecutive siblings unless a # break occurs.
2. **# Reset Mechanism:**
    - On encountering #, sibling linking information is cleared.
    - The subsequent node after # connects only to its parent, not its previous sibling.
    - After placing that node, sibling linking resumes normally for subsequent nodes.
3. **Graph Rendering:**
    - The code uses `pygraphviz` for hierarchical layout (top-to-bottom) and `matplotlib` for visualization.
    - Each node displays a rank symbol.
    - Edges are drawn without arrowheads, focusing on hierarchy and sibling structure rather than direction.
4. **Checking Order with `check_string_allowed`:**

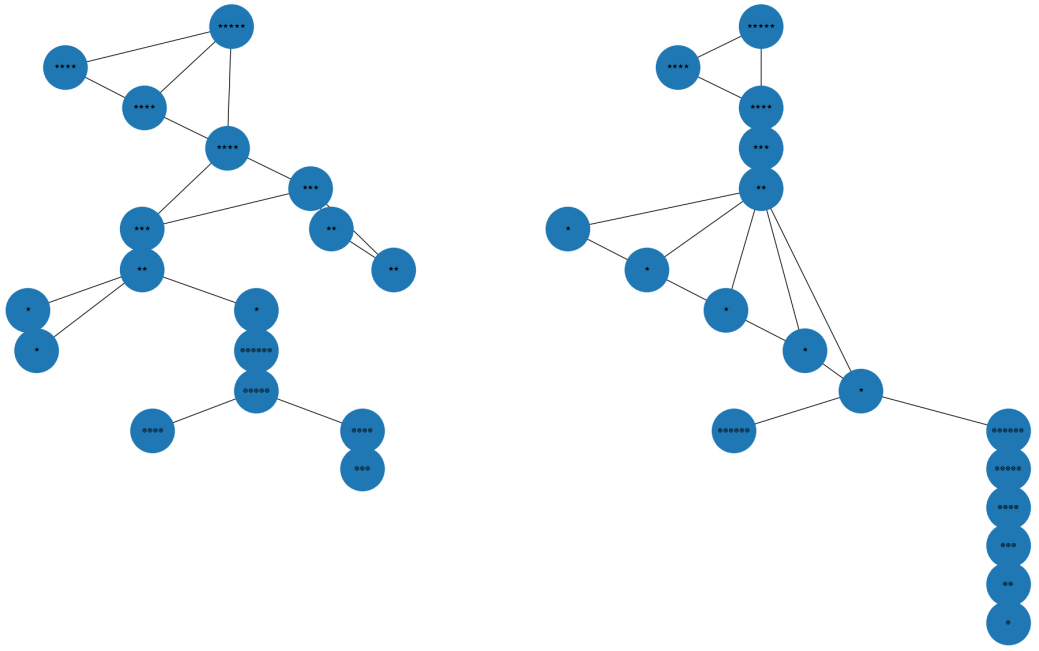

Figure 2: Example of the structure for input "ABBBCDD-CDEE#EFGH#HIABBCDEEEEEF#FGH#I#J#K"

- Before building the graph, you can run `check_string_allowed(input_string)` to ensure the string introduces ranks in proper alphabetical order of first appearance.
- Once all letters are introduced in the correct order, they can reappear in any sequence.

3.5 Example With Visualizations

- Consider an input "ABBCABB":
  - A might represent the top rank (General of the Air Force).
  - B corresponds to the next rank (General, O-10).
  - Additional Bs connect under A and link to each other as siblings.
  - Later, encountering C would connect to the last B.
  - Another A introduces a new top-level node, and subsequent Bs under that A form another sibling chain.
- Adding a # (e.g., "AB#B") breaks the sibling chain, so the second B after # does not form a sibling link with the previous B.

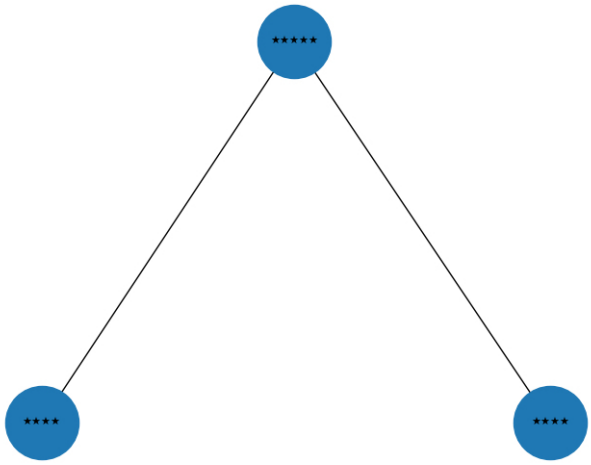

Figure 3: Example of the structure for input "AB#B"

4 Conclusion

This study presents a novel approach to hierarchical AI intelligence, inspired by human cognition and structured decision-making models like military rank structures. The Collective Intelligence DNA Model introduces a top-down control mechanism, where a DNA node assigns tasks, stem cell nodes collaborate and refine responses, and an aggregator node consolidates insights. Evaluations using OpenAI's Swarm framework with GPT-4 and Microsoft's AutoGen framework with LLaMA 3.2 (3B) highlight the model's potential to enhance multi-agent specialization and adaptability without modifying individual agents' pretrained structures.

A critical component of this research is the integration of Army Hierarchy principles to reinforce structured task delegation and decision-making. Just as military organizations rely on a clear chain of command for efficient coordination, the DNA Model applies a similar rank-based structure to AI agents. This ensures that complex queries are processed systematically, with specialized agents refining responses before a final decision is made at the top level. By embedding a hierarchical system into the AI framework, this research demonstrates how structured collaboration can reduce inefficiencies, improve response accuracy, and scale AI decision-making processes.

Future work will explore refining role assignments, optimizing hierarchical feedback loops, and integrating reinforcement learning to further enhance both multi-agent intelligence and structured AI governance. Ultimately, this research contributes to the advancement of decentralized, scalable, and adaptive AI systems, bridging the gap between isolated LLM performance and human-like hierarchical collaboration.

References

[1] Vernon B Mountcastle. The columnar organization of the neocortex. *Brain*, 120(4):701–722, 1997.

[2] Giulio Tononi and Gerald M Edelman. Consciousness and complexity. *Science*, 282(5395):1846–1851, 1998.

[3] OpenAI. Swarm framework documentation. 2023.

[4] Microsoft. Autogen documentation. 2023.

[5] Microsoft. Autogen documentation: Non-openai models - local ollama. `https://microsoft.github.io/autogen/0.2/docs/topics/non-openai-models/local-ollama`, 2023. Accessed: 2024-12-07.