

Scaling Large-Language-Model-based Multi-Agent Collaboration

Anonymous ACL submission

Abstract

Pioneering advancements in large language model-powered agents have underscored the design pattern of multi-agent collaboration, demonstrating that collective intelligence can surpass the capabilities of each individual. Inspired by the neural scaling law, which posits that increasing neurons leads to emergent abilities, this study investigates whether a similar principle applies to increasing agents in multi-agent collaboration. Technically, we propose multi-agent collaboration networks (MACNET), which utilize directed acyclic graphs to organize agents and streamline their interactive reasoning via topological ordering, with solutions derived from their dialogues. Extensive experiments show that MACNET consistently outperforms baseline models, enabling effective agent collaboration across various topologies. Notably, we observed a *small-world collaboration phenomenon*, where topologies resembling small-world properties achieved superior performance. Additionally, we identified a *collaborative scaling law*, indicating that normalized solution quality follows a logistic growth pattern as scaling agents, with collaborative emergence occurring much earlier than previously observed instances of neural emergence.

1 Introduction

In the rapidly advancing field of artificial intelligence, *large language models* (LLMs) have catalyzed transformative shifts across numerous domains due to their remarkable linguistic capacity to seamlessly integrate extensive world knowledge (Vaswani et al., 2017; Brown et al., 2020; Bubeck et al., 2023). Central to this breakthrough is the *neural scaling law* that fosters emergent capabilities, where well-trained neural networks often exhibit power-law scaling relations primarily with the number of neurons, alongside factors such as dataset size and training time (Kaplan et al., 2020; Schaeffer et al., 2024; Muennighoff et al., 2024).

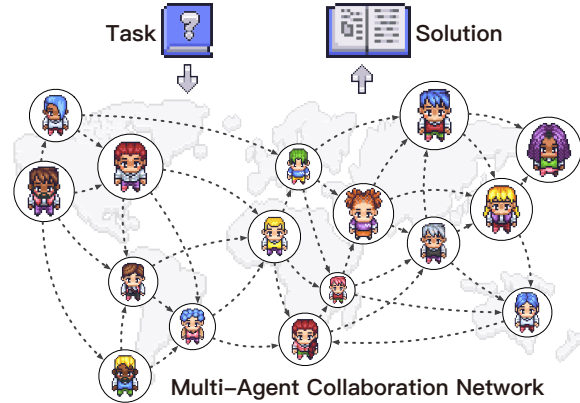


Figure 1: Given a task, multi-agent collaboration networks (MACNET) utilize directed acyclic graphs to organize diverse agents for collaborative interactions, with the final solution derived from their dialogues.

Despite this, LLMs have inherent limitations in enclosed reasoning, particularly when addressing complex situations that extend beyond textual boundaries (Richards, 2023). To this end, subsequent studies have successfully transformed foundational LLMs into versatile *autonomous agents* by equipping them with advanced capabilities such as tool use (Schick et al., 2023), long-context memory (Park et al., 2023), and procedural planning (Wei et al., 2022b). Along this line, *multi-agent collaboration* has emerged as an effective paradigm to integrate the specialties of different agents (Park et al., 2023; Li et al., 2023a; Qian et al., 2024b). Through linguistic interaction, agents engage in instructive and responsive utterances to foster high-quality collaboration, leading to final solutions¹ derived from their dialogues (Qian et al., 2024b,a; Chen et al., 2024b).

Inspired by the neural scaling law, a natural question arises: *does increasing agents in multi-agent collaboration exhibit emergent capabilities?*

¹Solutions can range from a multiple-choice answer to repository-level code or a coherent narrative, among numerous other possibilities.

Investigating the *collaborative scaling law* is essential for accurately estimating the relationship between computing resources and performance trends in multi-agent systems. This understanding enables optimizing resource utilization and minimizing unnecessary waste, ultimately leading to more scalable, practical, and resource-efficient agent systems (Kaplan et al., 2020). However, technically, effective multi-agent collaboration transcends the mere aggregation of responses from different agents through majority voting (Chen et al., 2024a); instead, it should constitute an organically integrated system that requires effective interactions and thoughtful decision-making (Hopfield, 1982).

In this paper, as illustrated in Figure 1, we envision multiple agents as a well-organized team composed of specialized agents, investigating their interdependent interactive reasoning and collective intelligence in autonomously solving complex problems. To further this goal, we design appropriate topologies and effective interaction mechanisms that align with both the static organizational structure and the dynamic reasoning process.

- To ensure generalizability, we design the topology as a directed acyclic graph where each edge is managed by a supervisory instructor issuing directional commands, and each node is supported by an executive assistant providing tailored solutions. This mechanism effectively fosters a division of labor among agents through functional dichotomy, seamlessly integrating a static topology with specialized agents to form a *multi-agent collaboration network* (MACNET).
- To facilitate agents’ interactive reasoning, the interaction sequence is orchestrated via topological ordering, ensuring orderly information transmission throughout the network. Within this arrangement, each interaction round involves two adjacent agents refining a previous solution, with only the refined solution, rather than the entire dialogue, being propagated to the next neighbors. This mechanism strategically avoids global broadcasts and significantly reduces the risk of overly extended contexts, enabling scalable collaboration across nearly any large-scale network.

We conducted a comprehensive quantitative evaluation of three prevalent topologies—chain, tree, and graph—divided into six variants, across multiple heterogeneous downstream scenarios. The

extensive experiments demonstrate that MACNET consistently outperforms all baseline models, enabling effective agent collaboration even in fully-connected dense networks. Notably, we observed a *small-world collaboration phenomenon*, where topologies resembling small-world properties demonstrated superior performance. Additionally, we identified a *collaborative scaling law*, revealing that normalized solution quality follows a logistic growth pattern as scaling agents. Meanwhile, collaborative emergence can be observed occurring significantly earlier compared to previous instances of neural emergence. We hope our findings offer valuable insights into resource prediction and optimization to enhance the efficiency and scalability of LLM systems (Kaplan et al., 2020).

2 Related Work

Trained on vast datasets and capable of manipulating billions of parameters, LLMs have become pivotal in natural language processing due to their seamless integration of extensive knowledge (Brown et al., 2020; Bubeck et al., 2023; Vaswani et al., 2017; Radford et al., 2019; Touvron et al., 2023; Wei et al., 2022a; Shanahan et al., 2023; Chen et al., 2021; Brants et al., 2007; Chen et al., 2021; Ouyang et al., 2022; Yang et al., 2024; Qin et al., 2023; Kaplan et al., 2020). Central to this breakthrough is the neural scaling law, which posits that loss scales as a power law with model size, dataset size, and the amount of compute used for training (Kaplan et al., 2020; Smith et al., 2022; Muennighoff et al., 2024). The principle underscores that scaling up language models can lead to emergent abilities—where performance experiences a sudden leap as the model scales (Wei et al., 2022a; Schaeffer et al., 2024).

Despite these, LLMs have inherent limitations in enclosed reasoning, motivating subsequent studies to effectively equip LLMs with advanced capabilities such as role playing (Li et al., 2023a; Chan et al., 2024), tool use (Schick et al., 2023; Qin et al., 2024), long-context memory (Park et al., 2023; Wang et al., 2023), and procedural planning (Wei et al., 2022b; Yao et al., 2023), thereby transforming fundamental LLMs into versatile autonomous agents (Richards, 2023; Shinn et al., 2024; Zhao et al., 2024). Along this line, multi-agent collaboration has emerged as an effective paradigm to integrate the specialities of different agents (Park et al., 2023; Zhou et al., 2023; Chen

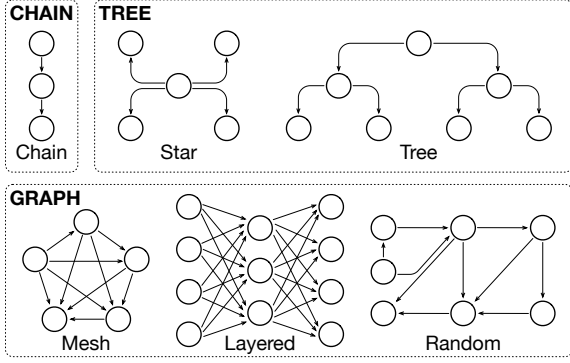


Figure 2: Representative topological structures.

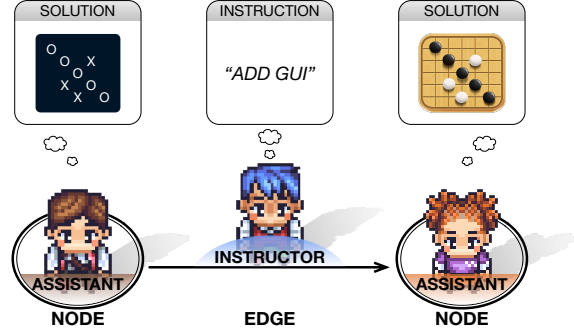


Figure 3: Assign different agents on nodes and edges.

et al., 2024b; Chan et al., 2024; Chen et al., 2023; Cohen et al., 2023; Li et al., 2023b; Hua et al., 2023). A straightforward collaboration strategy is majority voting (Chen et al., 2024a), where individuals remain independent; however, more effective multi-agent collaboration should form an integrated system that fosters interdependent interactions and thoughtful decision-making (Li et al., 2024; Chen et al., 2024a; Piatti et al., 2024). Based on this, pioneering studies have dichotomized the functionality of agents into two distinct roles: instructors, who provide directional instructions, and assistants, who respond with tailored solutions; these agents engage in instructive and responsive utterances to foster an interaction chain, collaboratively arriving at final solutions derived from their dialogues (Qian et al., 2024b; Li et al., 2023a). This paradigm facilitates a well-orchestrated workflow for task-oriented interactions, significantly reducing the need for manual intervention while demonstrating promising quality (Chen et al., 2024b; Chan et al., 2024).

3 Multi-Agent Collaboration Network

We aim to establish a scalable framework for multi-agent collaboration, comprising two key components: the design of a multi-agent collaboration network (MACNET) and collaborative reasoning.

3.1 Network Construction

To establish an organizational structure for multi-agent collaboration that is both efficient and scalable, drawing on the concept of graphs—a data structure that describes entities and their interrelations, we model the topology as a directed acyclic graph (DAG) (Nilsson et al., 2020) to organize interactions among collaborative agents (Qian et al., 2024a). Concretely, a feasible topology is denoted

as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$:

$$\mathcal{V} = \{v_i | i \in I\}, \mathcal{E} = \{(v_i, v_j) | i, j \in I \wedge i \neq j\} \quad (1)$$

where \mathcal{V} denotes the set of nodes indexed by I , and \mathcal{E} denotes the set of edges, with each edge directed from one node to another and no cycles exist.

Given the impracticality of enumerating all possible topologies, our study focuses on three prevalent types—chain, tree, and graph—further divided into six structures, as depicted in Figure 2. Chain topologies, resembling the waterfall model (Petersen et al., 2009), linearly structuring interactions along agents. Tree topologies enable agents to branch out, interacting in independent directions; further categorized into "wider" star-shaped and "deeper" tree-shaped structures. Graph topologies support arbitrary interaction dependencies, with nodes having multiple children and parents, forming either divergent or convergent interactions; further classified into fully-connected mesh structures, MLP-shaped layered structures, and irregular random structures. These representative topologies are extensively studied in complex network (Strogatz, 2001; Albert and Barabási, 2001) and LLM agent reasoning (Liu et al., 2023; Besta et al., 2024), ensuring a comprehensive examination of the most significant and practical structures in understanding multi-agent systems.

In the ecosystem of LLM-powered agents, a functional dichotomy (Li et al., 2023a)—consisting of supervisory instructors who issue directional instructions and executive assistants who provide tailored solutions—can effectively promote division of labor, stimulate functional behaviors, and facilitate efficient task resolution (Qian et al., 2024b,a). To integrate this strategy into the topology, as depicted in Figure 3, we strategically assign an in-

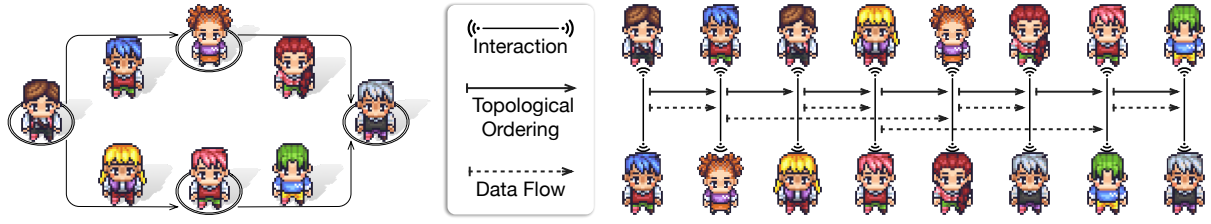


Figure 4: Streamlining the agents' reasoning process involves a series of dual-agent interactions. The topological order guides the interaction sequence, while the original connectivity governs the data flow.

235 structor to each edge and an assistant to each node:

$$236 \quad \begin{aligned} \mathbf{a}_i &= \rho(v_i), \forall v_i \in \mathcal{V} \\ \mathbf{a}_{ij} &= \rho(\langle v_i, v_j \rangle), \forall \langle v_i, v_j \rangle \in \mathcal{E} \end{aligned} \quad (2)$$

237 where $\rho(x)$ represents the agentization operation on
 238 an element x , achieved by equipping a foundation
 239 model with professional roles (Li et al., 2023a), ex-
 240 ternal tools (Schick et al., 2023), and context-aware
 241 memory (Park et al., 2023); \mathbf{a}_i and \mathbf{a}_{ij} denote an
 242 assistant agent assigned to node v_i and an instructor
 243 agent assigned to edge v_{ij} , respectively.

244 This dichotomous design allows agents to spe-
 245 cialize in their functions, driving task-oriented lan-
 246 guage interactions and facilitating efficient infor-
 247 mation transmission throughout the network. Addi-
 248 tionally, the "directed" nature of the edges enables
 249 the orchestration of agent interactions, while the
 250 "acyclic" configuration prevents information prop-
 251 agation deadlocks.

252 3.2 Interactive Reasoning

253 In the process of completing complex tasks, in-
 254 teractive reasoning among agents within a static
 255 MACNET requires strategical traversal to establish
 256 an orderly interaction sequence. Our graph trav-
 257 ersal strategy adheres to the principles of topological
 258 ordering (Bondy and Murty, 1976), a fundamental
 259 algorithm in graph theory, which ensures that each
 260 node is visited only after all its dependencies have
 261 been traversed (Gross et al., 2018). Formally, for
 262 a MACNET \mathcal{G} , its topological order is a linear ar-
 263 rangement of agents \mathbf{a}_i and \mathbf{a}_{ij} such that for every
 264 directed edge $\langle v_i, v_j \rangle \in \mathcal{E}$, the ordering satisfies:

$$265 \quad \forall \langle v_i, v_j \rangle \in \mathcal{E}, \mathbb{I}(\mathbf{a}_i) < \mathbb{I}(\mathbf{a}_{ij}) < \mathbb{I}(\mathbf{a}_j) \quad (3)$$

266 where $\mathbb{I}(x)$ denotes the index of agent x in the topo-
 267 logical sequence. This arrangement ensures that
 268 each node-occupied agent \mathbf{a}_i precedes its corre-
 269 sponding edge-occupied agent \mathbf{a}_{ij} , and \mathbf{a}_{ij} pre-
 270 cedees \mathbf{a}_j , thereby guaranteeing ensuring orderly
 271 information transmission throughout the network.

272 After establishing the global order, as illustrated
 273 in Figure 4, we enable each pair of edge-connected
 274 adjacent agents to interact and exchange informa-
 275 tion. For a topology \mathcal{G} , the design result in a total
 276 deployment of $|\mathcal{V}| + |\mathcal{E}|$ agents and require $2|\mathcal{E}|$
 277 interaction rounds. Within each edge, the interaction
 278 pattern between assistants and instructors follows
 279 a multi-turn instruction-response sequence (Qian
 280 et al., 2024b):

$$281 \quad \begin{aligned} \tau(\mathbf{a}_i, \mathbf{a}_{ij}, \mathbf{a}_j) &= (\tau(\mathbf{a}_i, \mathbf{a}_{ij}), \tau(\mathbf{a}_{ij}, \mathbf{a}_j)) \\ \tau(\mathbf{a}_i, \mathbf{a}_{ij}) &= (\mathbf{a}_i \rightarrow \mathbf{a}_{ij}, \mathbf{a}_{ij} \rightsquigarrow \mathbf{a}_i) \circlearrowleft \\ \tau(\mathbf{a}_{ij}, \mathbf{a}_j) &= (\mathbf{a}_{ij} \rightarrow \mathbf{a}_j, \mathbf{a}_j \rightsquigarrow \mathbf{a}_{ij}) \circlearrowleft \end{aligned} \quad (4)$$

282 where \rightarrow symbolizes the act of instructing, \rightsquigarrow in-
 283 dicates the corresponding responding, and \circlearrowleft rep-
 284 represents the iterative nature of the process. Specifi-
 285 cally, \mathbf{a}_i requests feedback, \mathbf{a}_{ij} offers optimization
 286 suggestions and requests further refinement, and
 287 \mathbf{a}_j provides the refined solution. Thus, the agents
 288 associated with a single edge can effectively opti-
 289 mize a solution in one iteration.

290 Delving deeper, the topological ordering method-
 291 ically unfolds agent interactions into an interaction
 292 sequence, outlining the control flow² within a multi-
 293 agent collaboration process. Concurrently, the data
 294 flow within this process is consistent with the origi-
 295 nal dependencies connected by edges, ensuring
 296 that the flow of interacted information aligns with
 297 the inherent dependencies outlined in the topology.

298 3.3 Memory Control

299 In a multi-agent collaboration system, unrestrained
 300 context information exchange can lead to exces-
 301 sively long contexts, ultimately limiting scalabil-
 302 ity by supporting only a few agents. To address
 303 this, we adopt a heuristic mechanism (Qian et al.,

²Note that although the interaction order is unfolded as a sequence for visualization purposes only, certain substructures (e.g., star-structured topology) inherently support parallel processing, which is essential in enhancing the reasoning efficiency of practical systems.

2024b) to manage context visibility using short-term and long-term memory. Short-term memory captures the intra-interaction working memory during each dual-agent interaction, ensuring context-aware decision-making. Long-term memory maintains inter-interaction context continuity by transmitting only the final solutions derived from dialogues, not the entire conversational history. This approach ensures that the context of ancestor agents remains Markovian, with solutions propagated only from adjacent agents rather than from all previous dialogues. Consequently, it reduces the risk of context overload while preserving context continuity, thereby enabling scalable multi-agent collaboration across nearly any large-scale network.

Furthermore, an original solution propagating through the network undergoes continuous refinement, improving its quality over time. As solutions traverse the network, they either branch off at divergent nodes or aggregate at convergent nodes. Branching is achieved through parallel propagation, while merging from multiple nodes, akin to a non-linear perceptron, requires an effective aggregation mechanism. Technically, convergent agents assess the strengths and weaknesses of each solution, synthesizing their strengths and discarding weaknesses, which results in a strength-aggregated outcome from "non-linear" decision-making, rather than a simple combination of all solutions.

4 Experiments

Baselines We select different kinds of representative methods for quantitative comparison.

- CoT (Wei et al., 2022b) is a technically general and empirically powerful method that endows LLMs with the ability to generate a coherent series of intermediate reasoning steps, naturally leading to the final solution through thoughtful thinking and allowing reasoning abilities to emerge.
- AUTOGPT (Richards, 2023) is a versatile single-agent system that employs multi-step planning and tool-augmented reasoning to autonomously decompose complex tasks into chained subtasks and iteratively leverages external tools within an environment-feedback cycle to progressively develop effective solutions.
- GPTSWARM (Zhuge et al., 2024) formalizes a swarm of LLM agents as computational graphs, where nodes represent manually customized functions and edges represent information flow, significantly surpassing the tree-of-thought method by optimizing node-level prompts and modifying graph connectivity.
- AGENTVERSE (Chen et al., 2024b) recruits and orchestrates a team of expert agents in either a horizontal or vertical topological structure, employing multi-agent linguistic interaction to autonomously refine solutions and demonstrating emergent performance compared to individual agents, serving as both general and powerful multi-agent framework.

Datasets and Metrics We adopt publicly available and logically challenging benchmarks to evaluate across heterogeneous downstream scenarios.

- MMLU (Hendrycks et al., 2020) provides a comprehensive set of logical reasoning assessments across diverse subjects and difficulties, utilizing multiple-option questions to measure general world knowledge and logical inference capabilities. We assess the quality of generated solutions via *accuracy*, reflecting the correctness of responses to multiple-choice questions.
- HumanEval (Chen et al., 2021), a widely recognized benchmark for function-level code generation, designed for measuring basic programming skills. We assess via *pass@k*, reflecting function correctness across multiple standard test cases.
- SRDD (Qian et al., 2024b) integrates complex textual software requirements from major real-world application platforms, designed for repository-level software development, including requirement comprehension, system design, and integration testing. We assess using *quality*, a comprehensive metric that integrates crucial factors including completeness, executability, and consistency.
- CommonGen-Hard (Madaan et al., 2023) requires models to generate coherent sentences incorporating discrete concepts, designed to test systems' advanced commonsense reasoning, contextual understanding, and creative problem-solving. We assess using a comprehensive score that integrates crucial factors including grammar, fluency, context relevance, and logic consistency (Li et al., 2018; Chen et al., 2024b).

Implementation Details By default, our method utilizes the GPT-3.5-turbo model, chosen for its optimal balance of reasoning efficacy and efficiency.













Method	Paradigm	MMLU	HumanEval	SRDD	CommonGen	AVG.
CoT		0.3544 [†]	<u>0.6098</u> [†]	0.7222 [†]	0.6165	0.5757 [†]
AUTOGPT		0.4485 [†]	0.4809 [†]	0.7353 [†]	0.5972 [†]	0.5655 [†]
GPTSWARM		0.2368 [†]	0.4969	0.7096 [†]	0.6222 [†]	0.5163 [†]
AGENTVERSE		0.2977 [†]	0.7256 [†]	0.7587 [†]	0.5399 [†]	0.5805
MACNET-CHAIN		0.6632	0.3720	0.8056	0.5903	0.6078
MACNET-STAR		0.4456	0.5549	0.7679	<u>0.7382</u>	0.6267
MACNET-TREE		0.3421	0.4878	0.8044	0.7718	0.6015
MACNET-MESH		<u>0.6825</u>	0.5122	0.7792	0.5525	<u>0.6316</u>
MACNET-LAYERED		0.2780	0.4939	0.7623	0.7176	0.5629
MACNET-RANDOM		0.6877	0.5244	<u>0.8054</u>	0.5912	0.6522

Table 1: The overall performance of LLM-driven methods across various datasets, including both single-agent () and multi-agent () paradigms. For each dataset, the highest scores are highlighted in bold, while the second-highest scores are underlined. A dagger (†) denotes statistically significant differences ($p \leq 0.05$) between the baseline and our chain-structured setting.

We enhance the diversity of perspectives by leveraging GPT-4 to generate a pool of 4,000 profiles for assignment. These agents are equipped to autonomously use external tools (*e.g.*, Python compilers), and their temperatures decrease linearly from 1.0 to 0.0 according to topology depths. During agent interactions, a maximum of three rounds of utterances is allowed. To ensure fairness, all baselines adhere to identical hyperparameters and settings in the evaluation. All code and data will be publicly available.

4.1 Does Our Method Lead to Superior Performance?

We first employ the simplest topology—chain—as the default setting for our comparative analysis. As shown in Table 1, the chain-structure method consistently outperforms all baseline methods across most metrics, demonstrating a significant margin of improvement. The primary advantage of MACNET, compared to a single agent providing answers from a specific perspective, lies in its facilitation of a sequential process where solutions are continuously refined. This enables autonomous and incremental optimization, effectively alleviating previously imperfect solutions or false hallucinations (Qian et al., 2024b,a; Chen et al., 2024b; Chan et al., 2024). Moreover, we observe that CoT exhibits strong performance on certain datasets, even surpassing some multi-agent methods in specific cases. This is primarily because the underlying knowledge of widely-researched benchmarks is largely embedded in foundational models, giving single agents

a notable capability in these relatively "simple" tasks. Although GPTSWARM self-organizes agents through dynamic optimization of nodes and edges, it still requires extensive task-specific customization for all agents and their behaviors, making it challenging to seamlessly transfer to heterogeneous downstream tasks. Given the increasing need for highly performant and automatic real-world systems, it is unrealistic to expect that all preparatory knowledge can be fully pre-encoded in foundation models, nor can specific adaptations be pre-made for all unforeseen complex tasks. Luckily, MACNET addresses this challenge by automatically generating various networks through simple hyperparameters (*e.g.*, topology type and scale), without requiring additional specific adaptations, which represents a more promising paradigm for enhancing autonomy, scalability, and generalizability.

In addition, we ablate agents' profiles and temperature, which roughly regresses to the graph-of-thought reasoning thoughts by a single agent who lacks a profile and has a temperature set to 0. We find that ablating these mechanisms results in significant performance degradation across all topologies, with an average decrease of 2.69%. This highlights the superior collective intelligence over any form of reasoning by a single agent, as the latter corresponds to a feature dimension reduction of the high-dimensional multi-agent combination space, which solidifies reasoning ability due to the lack of flexibility to explore a better configuration.

4.2 How Do Different Topologies Perform Against Each Other?

To understand the topological properties, we conducted extensive experiments by altering MACNET’s topologies. The results in Table 1 demonstrate that different topologies exhibit varying levels of effectiveness for distinct tasks. For instance, a chain topology is more suitable for software development, while a mesh topology excels in logical selection. No single topology consistently delivers optimal results across all tasks. Further observation reveals that topologies approaching the small-world property (Watts and Strogatz, 1998)—characterized by a small *average path length*³—tend to exhibit superior performance, which we refer to as the “*small-world collaboration phenomenon*”. Concretely, as each edge in MACNET triggers agent interactions, the graph’s density naturally represents the agents’ interaction density. Empirically, higher interaction density is associated with improved performance among the three coarse-grained topological types.⁴ This performance discrepancy can be attributed to the fact that a higher graph density generally correlates with a higher *clustering coefficient*⁵. This increase in the clustering coefficient results in more adjacent node pairs, decreasing the average path length; consequently, the likelihood of long-distance solution invisibility is correspondingly decreased. Along this reason, we also discover that irregular random structures outperform regular mesh structures. This advantage can be attributed to random edge connections, which, in analogy to social networks, potentially link “unacquainted” agents via a direct shortcut, making them adjacent “acquaintances” and implicitly reducing the average path length, thus resembling small-world properties. Meanwhile, unlike mesh topology, which exhibits the highest interaction density, random topology achieves an optimal balance between reduced arrangement depth and enhanced reasoning efficiency, making it a more suitable tradeoff in practice.

Additionally, it is observed that, given the same density, “wider” star-shaped topologies generally

³Average path length (Albert and Barabási, 2001) is the average number of steps along the shortest paths for all possible pairs of network nodes, which is a measure of the efficiency of information transport on a network.

⁴For example, the densely connected mesh topology outperforms the moderately dense tree topology, which in turn outperforms the sparsely connected chain topology.

⁵The clustering coefficient measures how densely connected a node’s neighbors are to each other (Strogatz, 2001).

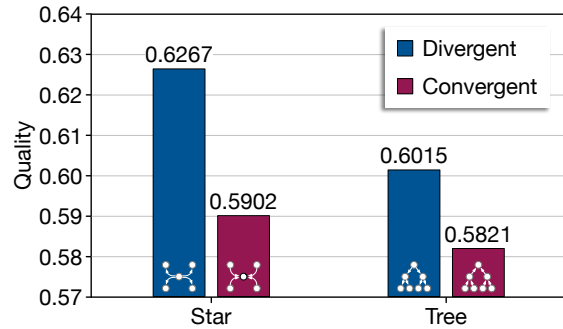


Figure 5: The average performance of the divergent topology (default) and its convergent counterpart.

outperform “deeper” tree-shaped ones. This is primarily due to our solution propagation mechanism, which inhibits the propagation of excessively long contextual reasoning processes throughout the entire network. As a result, deeper topologies may cause agents to lose sight of farther contexts, potentially leading to version rollback—solutions revert to earlier or similar versions. The same principle applies to graph structures, in which mesh topologies, compared to layered ones, enable direct reasoning between agents through direct edges, thereby implicitly reducing network depth and enhancing performance.

In addition to the structural point of view, the directional characteristics of some topologies, which exhibit inherent asymmetry—reversing the edges results in an entirely unequal one—motivated us to explore reverse topologies. As shown in Figure 5, merely altering the symmetry topologies’ orientation leads to significant performance degradation. Typically, divergent structures (those with more child nodes than parent nodes) significantly outperform convergent counterparts. Intuitively, solution flow smoothly diverges, allowing each agent to propose solutions from varied perspectives concurrently; conversely, converging the solutions of multiple agents at a single point poses a greater challenge, illustrating the complexity involved in integrating diverse perspectives into a cohesive strategy.

4.3 Does a Collaborative Scaling Law Exist?

Recall that the neural scaling law fosters emergent capabilities (Kaplan et al., 2020; Schaeffer et al., 2024; Muennighoff et al., 2024), where the synergy among numerous neurons enables a continuous trend of performance improvement. To investigate the *collaborative scaling law*—the potential predictable relationship between agent scale and

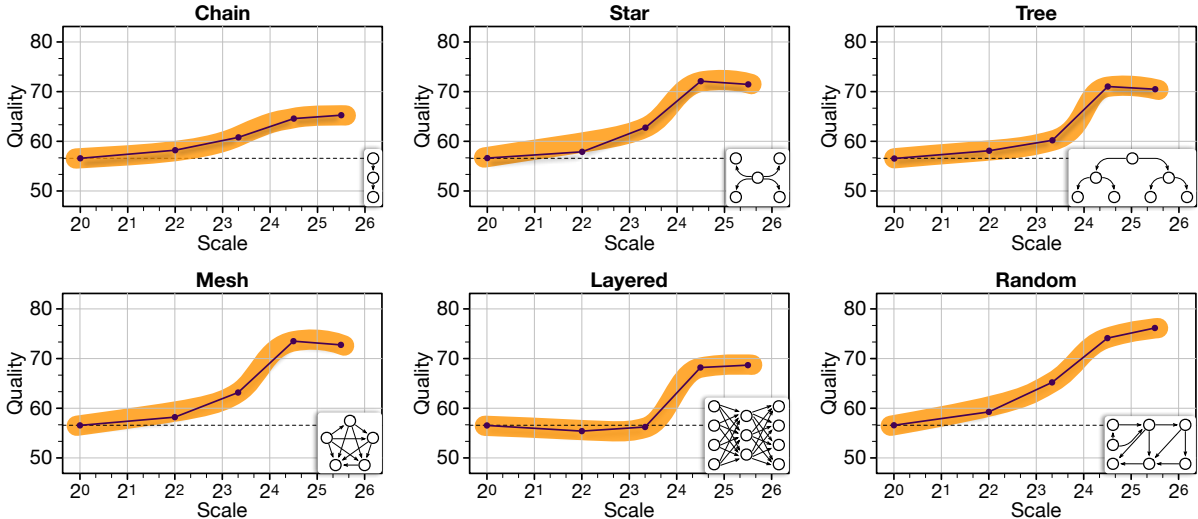


Figure 6: Scaling performance of multi-agent collaboration under different topologies. The topology scale is characterized by the number of nodes, which is distinct from the number of agents.

performance, considering the associated time and economic costs—we scaled different topologies by exponentially increasing the number of nodes from 1 (regressing to a single-agent method) to 50. As shown in Figure 6, our results confirm the small-world collaboration phenomenon, where optimal outcomes are achieved in high-density networks. Additionally, a reverse degradation phenomenon can be also observed (not shown), where certain configurations led to an overall quality reduction ranging from 2.27% to 6.24%.

As the topology scales, the quality of solutions produced by the multi-agent system initially rises rapidly before reaching a saturation point (or slightly declining), which can be approximated by a sigmoid-shaped function:

$$f(x) = \frac{\alpha}{1 + e^{-\beta(x-\gamma)}} + \delta \quad (5)$$

where α , β , γ and δ being real numbers specific to a topology. It is important to emphasize that this is only an average characterization based on the scale; a more precise multi-agent system should consider additional factors (*e.g.*, foundation models, profile, and tool spaces). Notably, neural scaling laws typically require a million-fold increase in neurons to reveal significant trends around a scale of 10^{18} to 10^{24} (Schaeffer et al., 2024). In contrast, most topologies in MACNET exhibit performance saturation around a scale of 2^4 to 2^5 . This collaborative emergence occurs more rapidly compared to neural emergence and is observable at smaller scales. The underlying reason is that neuron coordination, relying on from-scratch training in latent space

via matrix operations, requires a vast scale to incorporate extensive world knowledge and develop learning capabilities. In contrast, agent coordination, based on the implicit knowledge of pretrained LLMs, leverages the understanding and refinement of textual information through linguistic interactions, often circumventing the extensive scaling needed by neuronal coordination. Combining these two scaling mechanisms at different levels holds promise for producing higher-quality outcomes.

5 Conclusion

We have introduced MACNET, which leverages DAGs to structure the agents’ collaborative topologies and streamline their interactive reasoning through topological ordering, with solutions derived from their dialogues. Extensive experiments demonstrate that MACNET consistently outperforms all baseline models, enabling effective agent collaboration across various topologies. Notably, we observed a *small-world collaboration phenomenon*, where topologies resembling small-world properties demonstrated superior performance. Additionally, we identified a *collaborative scaling law*, revealing that normalized solution quality follows a logistic growth pattern as scaling agents. Meanwhile, collaborative emergence can be observed occurring significantly earlier compared to previous instances of neural emergence. We hope our findings offer valuable insights into resource prediction and optimization to enhance the efficiency and scalability of LLM systems.

6 Limitations

While our study has thoroughly explored the capabilities of collaborative autonomous agents across various tasks, it is crucial for both researchers and practitioners to remain cognizant of the limitations and risks associated with this study.

Compared to single-agent methods, the iterative interactions between multiple agents inherently demand more tokens and time, leading to increased computational requirements for the backbone models and potential environmental impacts. For example, our extensive experiment spanned more than six weeks and incurred costs of at least \$3,024.62. While the findings were informative and intriguing, the high resource expenditure raises concerns about the sustainability of future research endeavors. To address this, future research could focus on developing methods that enable agents to achieve equivalent or superior capabilities with fewer interactions. A promising strategy is to avoid full-graph inference by utilizing only a subset of the graph, such as identifying the best sub-team to execute the task.

We examined six representative topologies and identified a promising architectural direction through observed phenomena. However, within the vast space of network structures, identifying the theoretically optimal collaborative network of agents without bias remains a challenge. Further exploration into this optimal collaborative network is an interesting direction for future research. Moreover, there is significant value in exploring collaborative mechanisms, such as dynamically generating and deploying agents (including personalized profiles, external tools, multi-step planning, foundation models, and finer-grained labor division), and enhancing inference coordination (*e.g.*, efficient routing strategies, information transmission mechanisms, and long-context management).

In agents' reasoning, the aggregation of multiple solutions at graph nodes presents a complex challenge. The current strategy of combining strengths and eliminating weaknesses offers foundational insights but may fall short due to model hallucinations, potentially leading to performance degradation. We recommend designing the topology to minimize convergent nodes, while also developing a more robust aggregation strategy to significantly mitigate this issue.

The performance of multi-agent collaboration, given its additional factors, is inherently more un-

predictable than traditional scaling. We minimize bias through general designs and repeated experiments, but future work should consider more mature patterns (*e.g.*, usable tools) and higher-quality metrics. As current technology lacks precise automated evaluation systems for complex tasks (*e.g.*, software development and creative writing), manual verification becomes labor-intensive and impractical for large-scale datasets. This study focuses on objective and critical dimensions, such as comprehensive software indicators considering completeness, executability, and consistency. Future research should investigate finer-grained dimensions to enhance the objectivity and quantifiability of performance evaluations, including solutions' functionalities, robustness, safety, and user-friendliness.

Given the nascent stage of multi-agent collaboration models, most relevant studies focus on inference. When faced with diverse tasks, current methods handle each task independently due to the lack of methodologies that effectively incorporate past experiences. This inexperience often results in repetitive errors or unnecessary trial-and-error processes in multi-step tasks, requiring additional human intervention, especially in real-world applications. Therefore, multi-agent collaborative learning is an urgent area for research, promising more efficient cross-task inference and reduced resource consumption.

However, we believe that these potential limitations serve as inspiration for future research directions and can be effectively mitigated by engaging a broader, technically proficient audience. We expect that our findings will provide valuable insights into enhancing collaborative learning and reasoning in the ever-evolving dynamics of LLM-powered agents.

References

- Réka Albert and Albert-László Barabási. 2001. [Statistical Mechanics of Complex Networks](#). volume cond-mat/0106096.
- Maciej Besta, Florim Memedi, Zhenyu Zhang, Robert Gerstenberger, Guangyuan Piao, Nils Blach, Piotr Nyczyk, Marcin Copik, Grzegorz Kwaśniewski, Jürgen Müller, Lukas Gianinazzi, Ales Kubicek, Hubert Niewiadomski, Aidan O'Mahony, Onur Mutlu, and Torsten Hoefler. 2024. [Demystifying Chains, Trees, and Graphs of Thoughts](#). In *arXiv preprint arXiv:2401.14295*.

711	J. A. Bondy and U.R. Murty. 1976. Graph Theory with Applications . In <i>London: Macmillan</i> .	Roi Cohen, May Hamri, Mor Geva, and Amir Globerson. 2023. LM vs LM: Detecting Factual Errors via Cross Examination . In <i>Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)</i> , pages 12621–12640.	767
712			768
713	Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large Language Models in Machine Translation . In <i>Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)</i> , pages 858–867.	Jonathan L. Gross, Jay Yellen, and Mark Anderson. 2018. Graph Theory and Its Applications . In <i>Chapman and Hall/CRC</i> .	769
714			770
715			771
716			772
717			773
718			774
719			775
720	Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners . In <i>Advances in Neural Information Processing Systems (NeurIPS)</i> , volume 33, pages 1877–1901.	Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Xiaodong Song, and Jacob Steinhardt. 2020. Measuring Massive Multitask Language Understanding . In <i>arXiv preprint arXiv:2009.03300</i> .	776
721			777
722			778
723			779
724			780
725		J J Hopfield. 1982. Neural Networks and Physical Systems with Emergent Collective Computational Abilities . In <i>Proceedings Of The National Academy Of Sciences (PNAS)</i> .	781
726			782
727			783
728			784
729		Wenyue Hua, Lizhou Fan, Lingyao Li, Kai Mei, Jianchao Ji, Yingqiang Ge, Libby Hemphill, and Yongfeng Zhang. 2023. War and Peace (WarAgent): Large Language Model-based Multi-Agent Simulation of World Wars . In <i>arXiv preprint arXiv:2311.17227</i> .	785
730			786
731			787
732			788
733	Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of Artificial General Intelligence: Early Experiments with GPT-4 . In <i>arXiv preprint arXiv:2303.12712</i> .	Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models . In <i>arXiv preprint arXiv:2001.08361</i> .	789
734			790
735			791
736			792
737			793
738			794
739	Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2024. Chateval: Towards Better LLM-based Evaluators through Multi-agent Debate . In <i>The Twelfth International Conference on Learning Representations (ICLR)</i> .	Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023a. CAMEL: Communicative Agents for “Mind” Exploration of Large Language Model Society . In <i>Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)</i> .	795
740			796
741			797
742			798
743			799
744			800
745	Dake Chen, Hanbin Wang, Yunhao Huo, Yuzhao Li, and Haoyang Zhang. 2023. GameGPT: Multi-agent Collaborative Framework for Game Development . In <i>arXiv preprint arXiv:2310.08067</i> .	Junyou Li, Qin Zhang, Yangbin Yu, Qiang Fu, and Deheng Ye. 2024. More Agents is All You Need . <i>arXiv preprint arXiv:2402.05120</i> .	801
746			802
747			803
748			804
749	Lingjiao Chen, Jared Quincy Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James Zou. 2024a. Are More LLM Calls All You Need? Towards Scaling Laws of Compound Inference Systems . In <i>arXiv preprint arXiv:2403.02419</i> .	Yuan Li, Yixuan Zhang, and Lichao Sun. 2023b. Metaagents: Simulating Interactions of Human Behaviors for LLM-based Task-oriented Coordination via Collaborative Generative Agents . In <i>arXiv preprint arXiv:2310.06500</i> .	805
750			806
751			807
752			808
753			809
754	Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating Large Language Models Trained on Code . In <i>arXiv preprint arXiv:2107.03374</i> .	Zhongyang Li, Xiao Ding, and Ting Liu. 2018. Generating Reasonable and Diversified Story Ending using Sequence to Sequence Model with Adversarial Training . In <i>the International Conference on Computational Linguistics (COLING)</i> , pages 1033–1043.	810
755			811
756			812
757			813
758			814
759			815
760	Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chen Qian, Chi-Min Chan, Yujia Qin, Yaxi Lu, Ruobing Xie, et al. 2024b. AgentVerse: Facilitating Multi-agent Collaboration and Exploring Emergent Behaviors in Agents . In <i>The Twelfth International Conference on Learning Representations (ICLR)</i> .	Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. 2023. Dynamic LLM-Agent Network: An LLM-agent Collaboration Framework with Agent Team Optimization . In <i>arXiv preprint arXiv:2310.02170</i> .	816
761			817
762			818
763			819
764			820
765		Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang,	821
766			

822	Shashank Gupta, Bodhisattwa Prasad Majumder,	APIs. In <i>The Twelfth International Conference on</i>	879
823	Katherine Hermann, Sean Welleck, Amir Yazdan-	<i>Learning Representations (ICLR).</i>	880
824	bakhsh, and Peter Clark. 2023. Self-Refine: Iterative		
825	Refinement with Self-Feedback . In <i>Advances in Neu-</i>	Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang,	881
826	<i>ral Information Processing Systems (NeurIPS).</i>	Junru Wu, Jiaming Shen, Tianqi Liu, Jialu Liu, Don-	882
		ald Metzler, Xuanhui Wang, and Michael Bendersky.	883
827	Niklas Muennighoff, Alexander Rush, Boaz Barak,	2023. Large Language Models are Effective Text	884
828	Teven Le Scao, Nouamane Tazi, Aleksandra Pikt-	Rankers with Pairwise Ranking Prompting . In <i>arXiv</i>	885
829	tus, Sampo Pyysalo, Thomas Wolf, and Colin Raffel.	<i>preprint arXiv:2306.17563.</i>	886
830	2024. Scaling Data-Constrained Language Models .		
831	In <i>Advances in Neural Information Processing Sys-</i>	Alec Radford, Jeffrey Wu, Rewon Child, David Luan,	887
832	<i>tems (NeurIPS)</i> , volume 36.	Dario Amodei, Ilya Sutskever, et al. 2019. Language	888
		Models are Unsupervised Multitask Learners . In	889
833	Anton Nilsson, Carl Bonander, Ulf Strömberg, and	<i>OpenAI blog</i> , volume 1, page 9.	890
834	Jonas Björk. 2020. A Directed Acyclic Graph for		
835	Interactions . In <i>International Journal of Epidemiol-</i>	Toran Bruce Richards. 2023. AutoGPT . In	891
836	<i>ogy</i> .	https://github.com/Significant-Gravitas/AutoGPT .	892
837	Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida,	Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo.	893
838	Carroll Wainwright, Pamela Mishkin, Chong Zhang,	2024. Are Emergent Abilities of Large Language	894
839	Sandhini Agarwal, Katarina Slama, Alex Ray, John	Models a Mirage? In <i>Advances in Neural Informa-</i>	895
840	Schulman, Jacob Hilton, Fraser Kelton, Luke Miller,	<i>tion Processing Systems (NeurIPS)</i> , volume 36.	896
841	Maddie Simens, Amanda Askell, Peter Welinder,		
842	Paul F Christiano, Jan Leike, and Ryan Lowe. 2022.	Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta	897
843	Training Language Models to Follow Instructions	Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola	898
844	with Human Feedback . In <i>Advances in Neural In-</i>	Cancedda, and Thomas Scialom. 2023. Toolformer:	899
845	<i>formation Processing Systems (NeurIPS)</i> , volume 35,	Language Models Can Teach Themselves to Use	900
846	pages 27730–27744. Curran Associates, Inc.	Tools . In <i>arXiv preprint arXiv:2302.04761</i> .	901
847	Joon Sung Park, Joseph O’Brien, Carrie Jun Cai, Mered-	Murray Shanahan, Kyle McDonell, and Laria Reynolds.	902
848	ith Ringel Morris, Percy Liang, and Michael S Bern-	2023. Role Play with Large Language Models . In	903
849	stein. 2023. Generative Agents: Interactive Simu-	<i>Nature</i> , volume 623, pages 493–498.	904
850	lacra of Human Behavior . In <i>Proceedings of the 36th</i>		
851	<i>Annual ACM Symposium on User Interface Software</i>	Noah Shinn, Federico Cassano, Ashwin Gopinath,	905
852	<i>and Technology (UIST)</i> , pages 1–22.	Karthik Narasimhan, and Shunyu Yao. 2024. Reflex-	906
		ion: Language Agents with Verbal Reinforcement	907
853	Kai Petersen, Claes Wohlin, and Dejan Baca. 2009.	Learning . <i>Advances in Neural Information Process-</i>	908
854	The Waterfall Model in Large-Scale Development .	<i>ing Systems</i> , 36.	909
855	In <i>Product-Focused Software Process Improvement</i> ,		
856	pages 386–400.	Shaden Smith, Mostofa Patwary, Brandon Norick,	910
		Patrick LeGresley, Samyam Rajbhandari, Jared	911
857	Giorgio Piatti, Zhijing Jin, Max Kleiman-Weiner, Bern-	Casper, Zhun Liu, Shrimai Prabhumoye, George	912
858	hard Schölkopf, Mrinmaya Sachan, and Rada Mihal-	Zerveas, Vijay Korthikanti, Elton Zhang, Rewon	913
859	cea. 2024. Cooperate or Collapse: Emergence of	Child, Reza Yazdani Aminabadi, Julie Bernauer,	914
860	Sustainability Behaviors in a Society of LLM Agents .	Xia Song, Mohammad Shoeybi, Yuxiong He,	915
861	<i>arXiv preprint arXiv:2404.16698</i> .	Michael Houston, Saurabh Tiwary, and Bryan Catan-	916
		zaro. 2022. Using DeepSpeed and Megatron to	917
862	Chen Qian, Yufan Dang, Jiahao Li, Wei Liu, Zihao Xie,	Train Megatron-Turing NLG 530B, A Large-Scale	918
863	Yifei Wang, Weize Chen, Cheng Yang, Xin Cong,	Generative Language Model . In <i>arXiv preprint</i>	919
864	Xiaoyin Che, Zhiyuan Liu, and Maosong Sun. 2024a.	<i>arXiv:2201.11990</i> .	920
865	Experiential Co-Learning of Software-Developing		
866	Agents . In <i>the Annual Meeting of the Association for</i>	Steven H. Strogatz. 2001. Exploring Complex Net-	921
867	<i>Computational Linguistics (ACL)</i> .	works . In <i>Nature</i> .	922
868	Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan	Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier	923
869	Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng	Martinet, Marie-Anne Lachaux, Timothée Lacroix,	924
870	Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu,	Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal	925
871	and Maosong Sun. 2024b. ChatDev: Communicative	Azhar, et al. 2023. Llama: Open and Efficient	926
872	Agents for Software Development . In <i>the Annual</i>	Foundation Language Models . In <i>arXiv preprint</i>	927
873	<i>Meeting of the Association for Computational Lin-</i>	<i>arXiv:2302.13971</i> .	928
874	<i>guistics (ACL)</i> .		
		Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob	929
875	Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan	Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz	930
876	Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang,	Kaiser, and Illia Polosukhin. 2017. Attention is All	931
877	Bill Qian, et al. 2024. ToolLLM: Facilitating Large	You Need . In <i>Advances in Neural Information Pro-</i>	932
878	Language Models to Master 16000+ Real-World	<i>cessing Systems (NeurIPS)</i> , volume 30.	933

934 Haotian Wang, Xiyuan Du, Weijiang Yu, Qianglong
935 Chen, Kun Zhu, Zheng Chu, Lian Yan, and Yi Guan.
936 2023. [Apollo’s Oracle: Retrieval-Augmented Reasoning in Multi-Agent Debates](#). In *arXiv preprint arXiv:2312.04854*.
937
938

939 Duncan J. Watts and Steven H. Strogatz. 1998. [Collective Dynamics of Small-World Networks](#). *Nature*,
940 393:440–442.
941

942 Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel,
943 Barret Zoph, Sebastian Borgeaud, Dani Yogatama,
944 Maarten Bosma, Denny Zhou, Donald Metzler, Ed H.
945 Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy
946 Liang, Jeff Dean, and William Fedus. 2022a. [Emergent Abilities of Large Language Models](#). In *Transactions on Machine Learning Research*.
947
948

949 Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten
950 Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and
951 Denny Zhou. 2022b. [Chain-of-thought Prompting Elicits Reasoning in Large Language Models](#). In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 24824–24837.
952
953
954

955 Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao
956 Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen.
957 2024. [Large Language Models as Optimizers](#). In *The Twelfth International Conference on Learning Representations (ICLR)*.
958
959

960 Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom
961 Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. [Tree of Thoughts: Deliberate Problem Solving with Large Language Models](#). In *Advances in Neural Information Processing Systems (NeurIPS)*.
962
963
964

965 Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin,
966 Yong-Jin Liu, and Gao Huang. 2024. [Expel: LLM Agents are Experiential Learners](#). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19632–19642.
967
968
969

970 Wangchunshu Zhou, Yuchen Eleanor Jiang, Long Li,
971 Jialong Wu, Tiannan Wang, Shi Qiu, Jintian Zhang,
972 Jing Chen, Ruipu Wu, Shuai Wang, Shiding Zhu, Jiyu
973 Chen, Wentao Zhang, Ningyu Zhang, Huajun Chen,
974 Peng Cui, and Mrinmaya Sachan. 2023. [Agents: An Open-source Framework for Autonomous Language Agents](#). In *arXiv preprint arXiv:2309.07870*.
975
976

977 Mingchen Zhuge, Wenyi Wang, Louis Kirsch,
978 Francesco Faccio, Dmitrii Khizbullin, and Jurgen
979 Schmidhuber. 2024. [Language Agents as Optimizable Graphs](#). *arXiv preprint arXiv:2402.16823*.
980

981	Appendix	
982	A Ablation Study of Temperature	
983	In this appendix, an ablation study is meticulously	1027
984	conducted to investigate the effect of varied tem-	1028
985	perature (which, in this study, decreases linearly)	1029
986	of agents on the performance of MACNET. The	1030
987	temperature parameter for all agents was uniformly	1031
988	set to 0.0, thereby rendering the entire reasoning	1032
989	process deterministic. The experiments were exclu-	1033
990	sively carried out on scenarios where the number	1034
991	of nodes equals 10, as it is a moderate scale that	1035
992	features both low costs and a sufficient amount of	1036
993	multi-agent collaboration. To ensure consistency,	1037
994	the topologies and datasets in this study align with	1038
995	the configurations shown in Table 1.	1039
996	As depicted in Figure 7, the overall performance	1040
997	of MACNET exhibits an average deterioration of	1041
998	approximately 7.41% in the absence of a linearly	1042
999	decreasing temperature. Notably, the performance	1043
1000	of the star topology experiences a particularly sig-	1044
1001	nificant decline of about 15.5%. This phenomenon	1045
1002	suggests that the temperature deployment mecha-	1046
1003	nism of MACNET is effective.	1047
1004	B Ablation Study of Profiles	
1005	To study the role of profiles in the agent reason-	1048
1006	ing process within our system, we orchestrated a	1049
1007	series of experiments in which the profiles of all	1050
1008	agents were left blank. Except for the implementa-	1051
1009	tion of the linearly decreasing temperature, other	1052
1010	configurations are identical with Appendix 7.	1053
1011	As illustrated in Figure 8, the performance of	1054
1012	MACNET deteriorates for an average of 3.75% with	1055
1013	the absence of the profiles. This phenomenon sug-	1056
1014	gests that the profile deployment mechanism of	1057
1015	MACNET is effective.	1058
1016	C Performance analysis of Claude	
1017	We conducted experiments utilizing Claude ⁶ as the	1059
1018	base model. The number of nodes was set to 4 and	1060
1019	datasets were selected as SRDD and CommonGen,	1061
1020	mainly considering costs. Temperature, profile de-	1062
1021	ployment, and topologies align with the config-	1063
1022	urations delineated in implementation details in	1064
1023	section 4.	1065
1024	Figure 9 shows that the performance of Claude	1066
1025	is worse than gpt-3.5.	1067
	⁶ Claude 3 sonnet (until 20240229), by Anthropic.	1068
		1069
		1070
		1071
		1072
	D Time Consumption Analysis	
	To investigate the time costs of MACNET and the	1027
	underlying mechanisms, we analyzed the results	1028
	on the SRDD dataset. To maximize the difference	1029
	in topological properties (<i>e.g.</i> , graph density, max-	1030
	imum depth, etc.) the number of nodes is chosen	1031
	as 50. As mentioned in section 3.2, a topology \mathcal{G}	1032
	requires $2 \mathcal{E} $ interaction rounds. Therefore, for dif-	1033
	ferent types of topologies, their interaction rounds	1034
	can be calculated as in Figure 10. After carefully	1035
	examining the experiment logs, it can be concluded	1036
	that consumed time is positively correlated with	1037
	the quantity of interaction rounds. We recorded the	1038
	average time consumed on each type of topology,	1039
	as shown in Figure 11.	1040
	Similar results can also be obtained from other	1041
	datasets and topologies. Moreover, we noticed that	1042
	the cost increases exponentially as the number of	1043
	interaction rounds increases, instead of linearly.	1044
	Consequently, it is suggested that future imple-	1045
	mentation should carefully balance the cost and	1046
	performance.	1047
	E The MMLU Dataset	
	The MMLU dataset is a massive multitask test con-	1048
	sisting of multiple-choice questions from various	1049
	branches of knowledge. The test covers 57 tasks in-	1050
	cluding elementary mathematics, US history, com-	1051
	puter science, law, and more. It ranges in difficulty	1052
	from an elementary level to an advanced profes-	1053
	sional level, and it tests both world knowledge and	1054
	problem-solving ability. All 57 tasks and their de-	1055
	tailed topics are shown in Figure 12. The initial	1056
	format of questions is shown in Figure 13.	1057
	F The HumanEval Dataset	
	The HumanEval dataset comprises 164 hand-	1058
	written programming problems, each including a	1059
	function signature, a docstring, a function body,	1060
	and multiple unit tests. Problems are designed to	1061
	test the model’s ability to generate functionally cor-	1062
	rect code from natural language specifications. For	1063
	instance, the tasks often involve implementing al-	1064
	gorithms for sorting, searching, and manipulating	1065
	data structures such as arrays and strings. An ex-	1066
	ample of the initial prompt of the HumanEval test	1067
	is shown in Figure 14. Each problem also includes	1068
	multiple test cases that validate the correctness of	1069
	the generated code.	1070

1073
1074
1075
1076
1077
1078
1079

1080
1081
1082
1083
1084

1085
1086
1087
1088
1089

1090
1091
1092
1093
1094
1095
1096

1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121

G The SRDD Dataset

The SRDD Dataset is a comprehensive database containing 1,200 software descriptions for automatic software generation. The dataset structure is shown in Figure 15. The construction of this database adhered to the following three-stage strategy for constructing a diverse and unique dataset:

1. Random Sampling: First, ChatGPT is independently inquired multiple times to obtain software information under a certain category, and then the duplication is removed at the token granularity of the software name.
2. Sequential Sampling: Then we add the generated software information in sequence in the form of negative prompts, requiring ChatGPT to continue generating unique software information.
3. Check: Although ChatGPT has been required to follow certain rules when generating, LLM is more likely to be overconfident when generating according to rules than when judging based on rules. Therefore, our last step is to let ChatGPT determine whether the generated software follows the rules.

This strategy initially establishes datasets by random sampling some software data, then records existing data, granting ChatGPT autonomy to produce novel entries. SRDD is created with human-designed rules that make the created software easy for researchers to evaluate, for example, the collected software does not need internet or multi-player participation. The length distribution of software descriptions in SRDD is shown in Figure 16.

We sought to analyze the effects and semantic features of the generated software descriptions by using t-SNE to perform dimensionality reduction and visualization on the description embedding generated by the OpenAI Ada Model.

As demonstrated in figure 17, significant clustering of tasks bearing the same color is observed. It can be concluded that 1) software descriptions of the same category are distributed in clusters, indicating that the generated descriptions are highly related to their categories. 2) Descriptions in different subcategories under the same category are clustered together, such as the game subcategories in the lower right corner. 3) Some subcategories of different categories also show overlaps in the figure, such as Tools&Utilities and Graphics, Schedule and

Business, Sports and Sports Game. Such an overlap is comprehensible given the multi-functionality of some software applications that may not be confined to a single classification.

H The CommonGen-Hard Dataset

The CommonGen dataset is a constrained text generation task designed to evaluate the ability of generative models in commonsense reasoning. The dataset is composed of 35,141 unique concept sets and corresponding human-annotated sentences that describe everyday scenarios using those concepts.

The CommonGen-Hard dataset is a more challenging variant of the original dataset CommonGen. CommonGen-Hard requires models to generate coherent and grammatically correct sentences incorporating 20-30 concepts, as opposed to the original task which presents a set of 3-5 related concepts. This significant increase in the number of concepts tests the model's ability to perform advanced commonsense reasoning, contextual understanding, and creative problem-solving, as it must generate meaningful sentences that encompass a broader range of ideas.

Two key challenges of the tests are *rational reasoning* with underlying commonsense knowledge about given concepts, and *compositional generalization* for unseen combination of concepts. Samples shown in Figure 18 include a concept set and the coherent sentences generated by models.

I Case Study

This section presents a case study on software developed, detailing each stage of its lifecycle. The representative software is "Business Sales Performance Tracker" with a user's requirement: "Business Sales Performance Tracker is a software application that helps businesses track and analyze their sales performance. It provides features for inputting sales data, generating reports, and visualizing sales performance metrics. The application also allows businesses to set sales goals and compare actual performance against targets".

Figure 19 illustrates the Business Sales Performance Tracker's user interface. On the top left, a data entry interface is displayed, where users can input sales-related information. This interface allows for the repeated entry of customer names, product names, and sales figures into designated fields. Users can then click the "Add Sales Data" button to integrate this information into the tracking system.

1122
1123
1124
1125

1126

1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150

1151

1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170

1171 For generating comprehensive reports, the user can
1172 click the "Generate Report" button. This action pro-
1173 duces a statistical report within a terminal window,
1174 displaying key metrics such as total revenue, sales
1175 growth, conversion rate, average order value, cus-
1176 tomer acquisition cost, and customer lifetime value.
1177 Additionally, a visual report in the form of a his-
1178 togram is displayed on the right side of the window.
1179 The software includes tools in the toolbar, which
1180 enable users to customize the histogram's layout
1181 and style. These tools also provide options to save
1182 and export the graphical data representations.

1183 Figures 20, 21, 22, 23, 24, 25, 26, 27, and 28
1184 provide a comprehensive view of the multi-agent
1185 interaction. Each figure captures the detailed di-
1186 alogue and interactions, showcasing the collabo-
1187 rative efforts and methodologies employed in the
1188 development of the software.

1189 For screenshots of other examples of software
1190 that MACNET-CHAIN has been able to build, see
1191 Figure 29.

1192 **J License**

1193 The four datasets used in this experiment are all li-
1194 censed under the CC-BY-NC-4.0 license, allowing
1195 free use for scientific research.

1196 **K Software and Data**

1197 The source code of the system and the datasets uti-
1198 lized in this paper are provided in `Software.zip`
1199 and `Data.zip`, respectively. These archives pro-
1200 vide comprehensive configuration guidelines, com-
1201 mand instructions for execution, examples of logs,
1202 and additional resources. These components are
1203 essential for assessing the reproducibility of our
1204 technology. Moreover, all these materials will be
1205 made publicly available to support subsequent aca-
1206 demic research.

1207 **L AI Assistants**

1208 ChatGPT⁷ was used purely with the language of the
1209 paper during the writing process, including spell-
1210 checking and paraphrasing the authors' original
1211 content, without suggesting new content. Any con-
1212 tent generated with the assistant underwent meticu-
1213 lous manual review and subsequently received final
1214 approval from the authors.

⁷<https://chat.openai.com/>

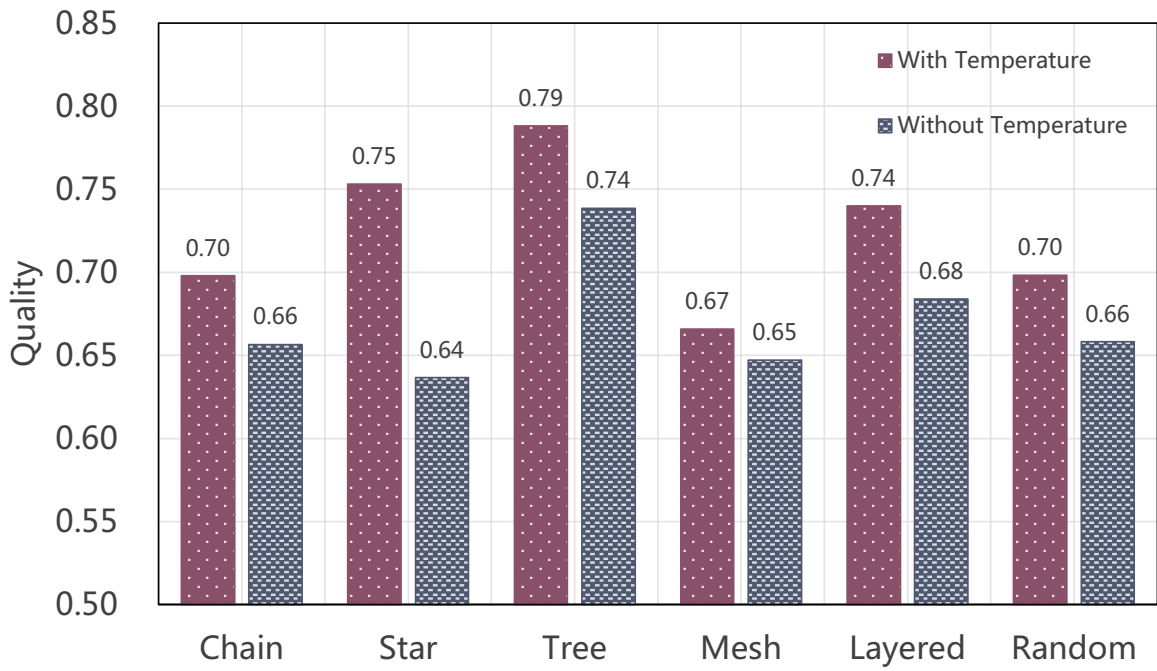


Figure 7: Ablation study on temperature under different topologies.

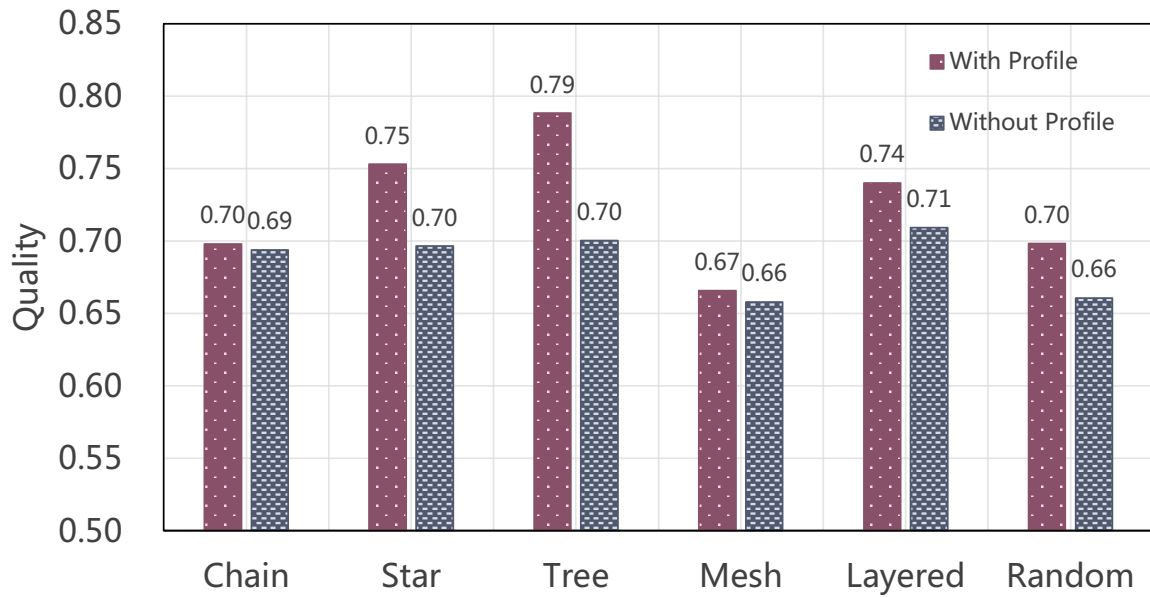


Figure 8: Ablation study on profiles under different topologies.

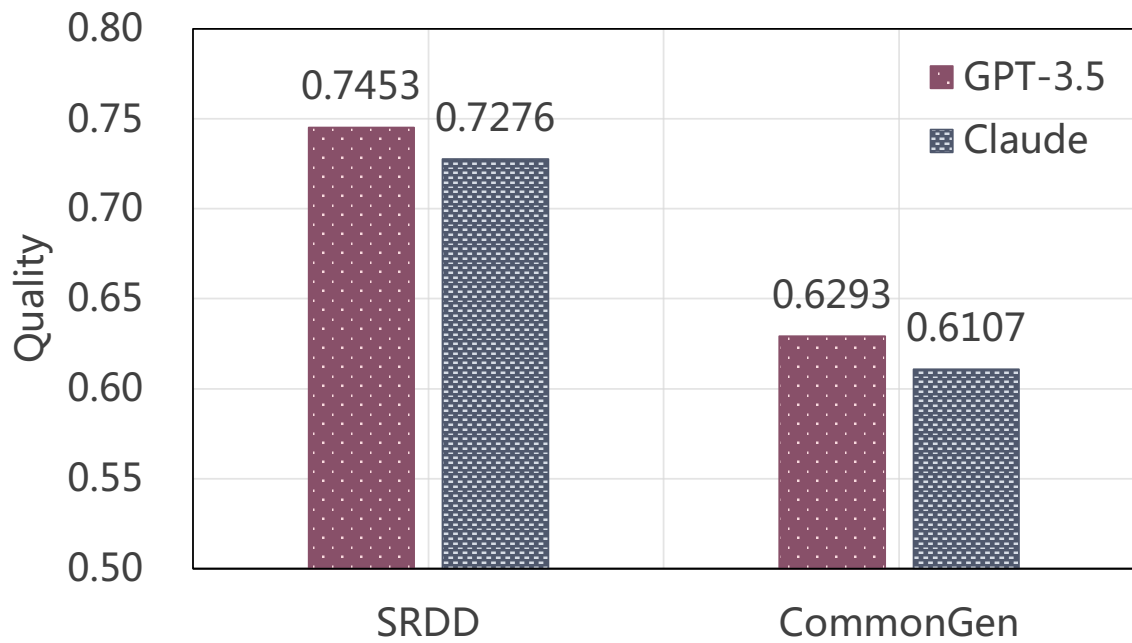


Figure 9: Performances of Claude and GPT-3.5 on SRDD and CommonGen-Hard datasets. Temperature linearly decreases from 1.0 to 0.0. Profiles are assigned from the pool.

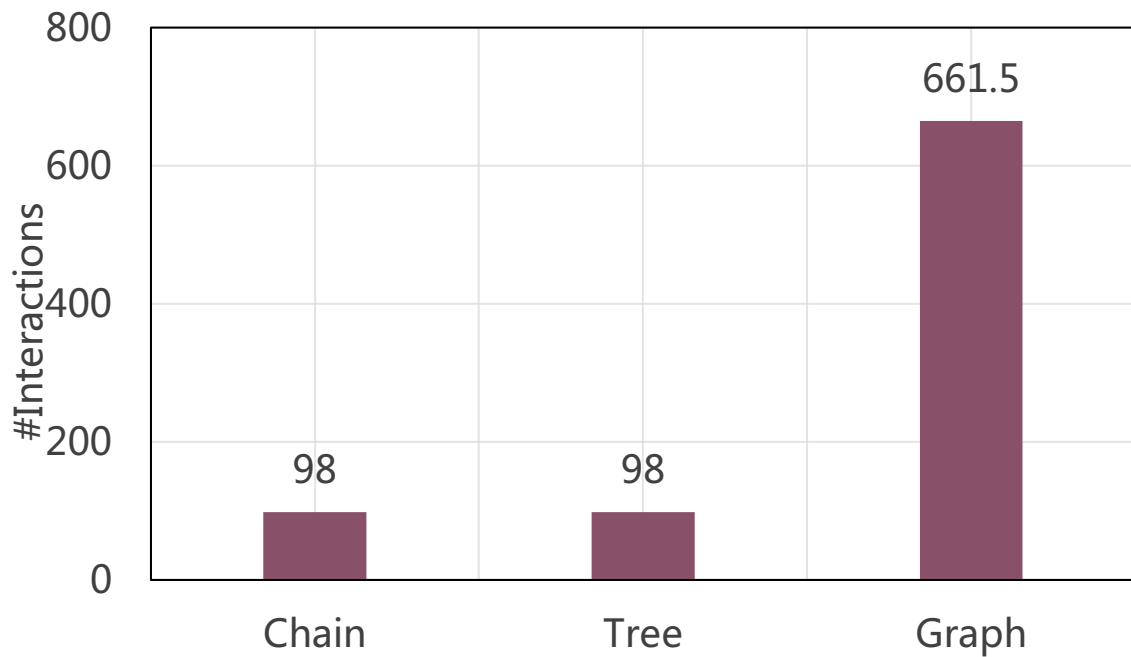


Figure 10: The quantity of interaction rounds in Chain, Tree, and Graph topologies. The number of nodes equals 50.

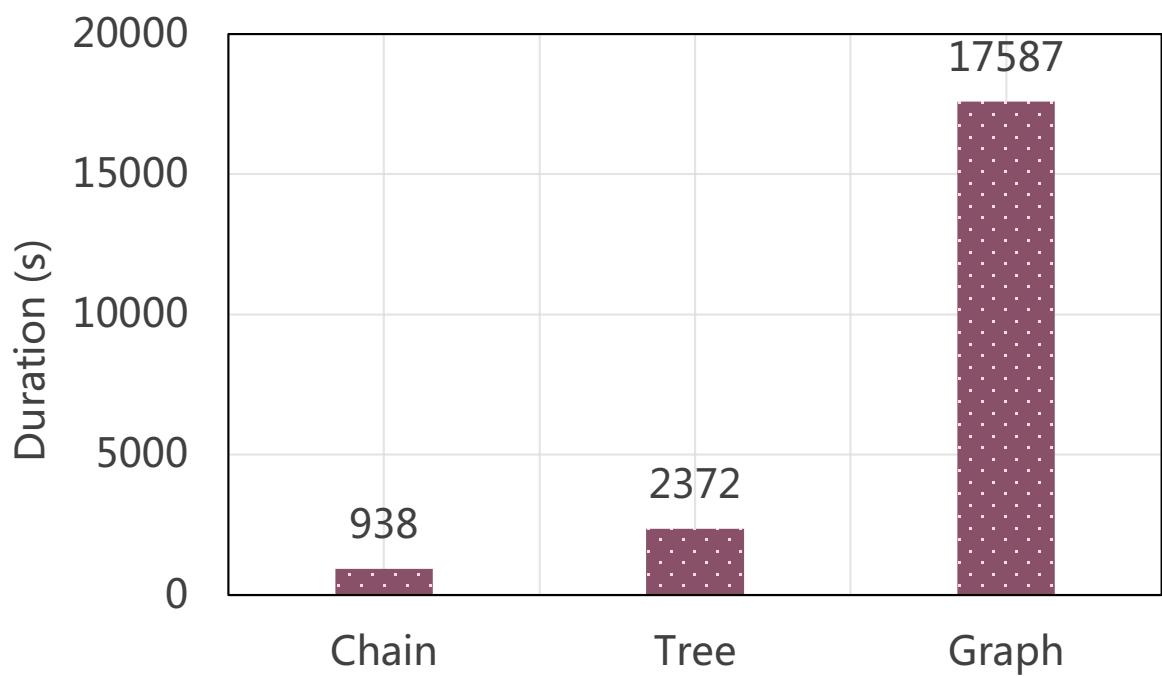


Figure 11: Average time consumed (duration) in experiments under different topologies. The number of nodes equals 50.

Task	Tested Concepts	Supercategory
Abstract Algebra	Groups, rings, fields, vector spaces, ...	STEM
Anatomy	Central nervous system, circulatory system, ...	STEM
Astronomy	Solar system, galaxies, asteroids, ...	STEM
Business Ethics	Corporate responsibility, stakeholders, regulation, ...	Other
Clinical Knowledge	Spot diagnosis, joints, abdominal examination, ...	Other
College Biology	Cellular structure, molecular biology, ecology, ...	STEM
College Chemistry	Analytical, organic, inorganic, physical, ...	STEM
College Computer Science	Algorithms, systems, graphs, recursion, ...	STEM
College Mathematics	Differential equations, real analysis, combinatorics, ...	STEM
College Medicine	Introductory biochemistry, sociology, reasoning, ...	Other
College Physics	Electromagnetism, thermodynamics, special relativity, ...	STEM
Computer Security	Cryptography, malware, side channels, fuzzing, ...	STEM
Conceptual Physics	Newton's laws, rotational motion, gravity, sound, ...	STEM
Econometrics	Volatility, long-run relationships, forecasting, ...	Social Sciences
Electrical Engineering	Circuits, power systems, electrical drives, ...	STEM
Elementary Mathematics	Word problems, multiplication, remainders, rounding, ...	STEM
Formal Logic	Propositions, predicate logic, first-order logic, ...	Humanities
Global Facts	Extreme poverty, literacy rates, life expectancy, ...	Other
High School Biology	Natural selection, heredity, cell cycle, Krebs cycle, ...	STEM
High School Chemistry	Chemical reactions, ions, acids and bases, ...	STEM
High School Computer Science	Arrays, conditionals, iteration, inheritance, ...	STEM
High School European History	Renaissance, reformation, industrialization, ...	Humanities
High School Geography	Population migration, rural land-use, urban processes, ...	Social Sciences
High School Gov't and Politics	Branches of government, civil liberties, political ideologies, ...	Social Sciences
High School Macroeconomics	Economic indicators, national income, international trade, ...	Social Sciences
High School Mathematics	Pre-algebra, algebra, trigonometry, calculus, ...	STEM
High School Microeconomics	Supply and demand, imperfect competition, market failure, ...	Social Sciences
High School Physics	Kinematics, energy, torque, fluid pressure, ...	STEM
High School Psychology	Behavior, personality, emotions, learning, ...	Social Sciences
High School Statistics	Random variables, sampling distributions, chi-square tests, ...	STEM
High School US History	Civil War, the Great Depression, The Great Society, ...	Humanities
High School World History	Ottoman empire, economic imperialism, World War I, ...	Humanities
Human Aging	Senescence, dementia, longevity, personality changes, ...	Other
Human Sexuality	Pregnancy, sexual differentiation, sexual orientation, ...	Social Sciences
International Law	Human rights, sovereignty, law of the sea, use of force, ...	Humanities
Jurisprudence	Natural law, classical legal positivism, legal realism, ...	Humanities
Logical Fallacies	No true Scotsman, base rate fallacy, composition fallacy, ...	Humanities
Machine Learning	SVMs, VC dimension, deep learning architectures, ...	STEM
Management	Organizing, communication, organizational structure, ...	Other
Marketing	Segmentation, pricing, market research, ...	Other
Medical Genetics	Genes and cancer, common chromosome disorders, ...	Other
Miscellaneous	Agriculture, Fermi estimation, pop culture, ...	Other
Moral Disputes	Freedom of speech, addiction, the death penalty, ...	Humanities
Moral Scenarios	Detecting physical violence, stealing, externalities, ...	Humanities
Nutrition	Metabolism, water-soluble vitamins, diabetes, ...	Other
Philosophy	Skepticism, phronesis, skepticism, Singer's Drowning Child, ...	Humanities
Prehistory	Neanderthals, Mesoamerica, extinction, stone tools, ...	Humanities
Professional Accounting	Auditing, reporting, regulation, valuation, ...	Other
Professional Law	Torts, criminal law, contracts, property, evidence, ...	Humanities
Professional Medicine	Diagnosis, pharmacotherapy, disease prevention, ...	Other
Professional Psychology	Diagnosis, biology and behavior, lifespan development, ...	Social Sciences
Public Relations	Media theory, crisis management, intelligence gathering, ...	Social Sciences
Security Studies	Environmental security, terrorism, weapons of mass destruction, ...	Social Sciences
Sociology	Socialization, cities and community, inequality and wealth, ...	Social Sciences
US Foreign Policy	Soft power, Cold War foreign policy, isolationism, ...	Social Sciences
Virology	Epidemiology, coronaviruses, retroviruses, herpesviruses, ...	Other
World Religions	Judaism, Christianity, Islam, Buddhism, Jainism, ...	Humanities

Figure 12: Tasks of the MMLU dataset.

MMLU Prompt

The following are multiple-choice questions (with answers) about abstract algebra.
Find the degree for the given field extension $Q(\sqrt{2}, \sqrt{3}, \sqrt{18})$ over Q .

A. 0

B. 4

C. 2

D. 6

Answer:

Figure 13: The official prompt of the MMLU dataset.

HumanEval Prompt

```
from typing import List
```

```
def below_zero(operations: List[int]) -> bool:
```

```
    """ You're given a list of deposit and withdrawal operations on a bank account that starts  
        with  
        zero balance. Your task is to detect if at any point the balance of account falls below  
        zero, and  
        at that point function should return True. Otherwise it should return False.
```

```
>>> below_zero([1, 2, 3])
```

```
False
```

```
>>> below_zero([1, 2, -4, 5])
```

```
True
```

```
"""
```

Figure 14: The official prompt of the HumanEval dataset.

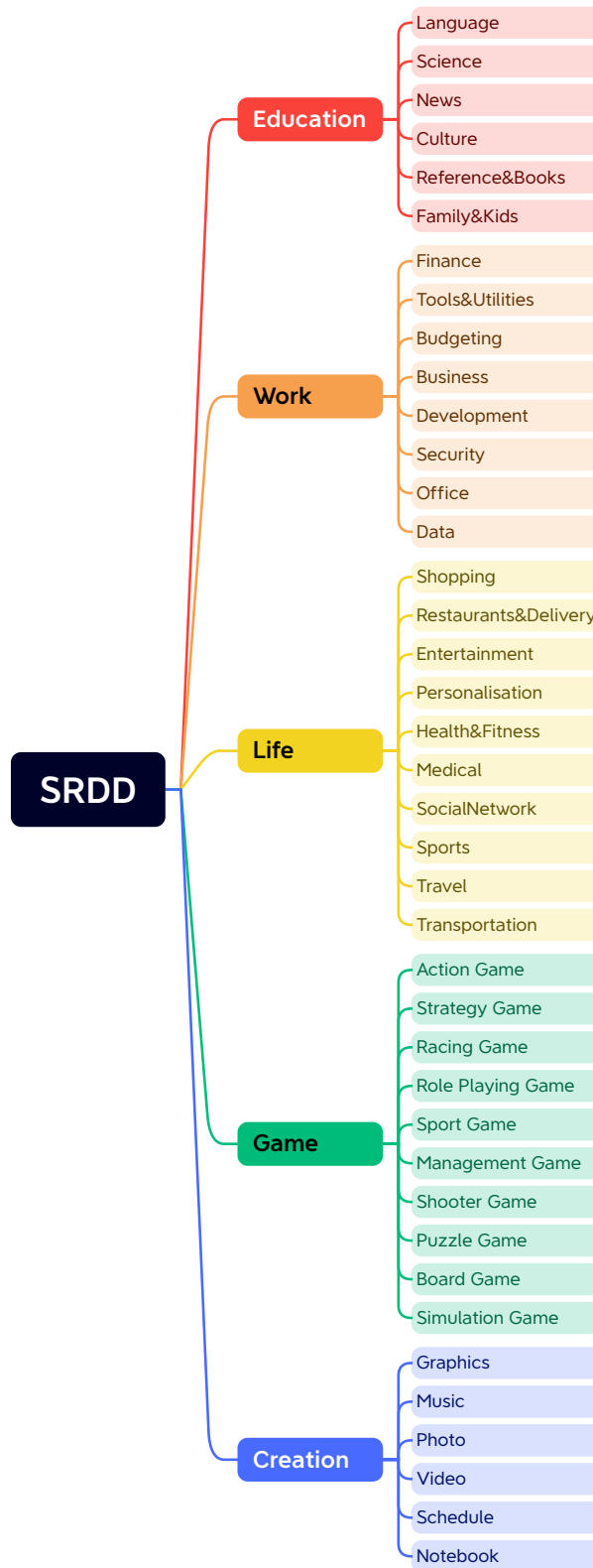


Figure 15: The hierarchy of the SRDD dataset.

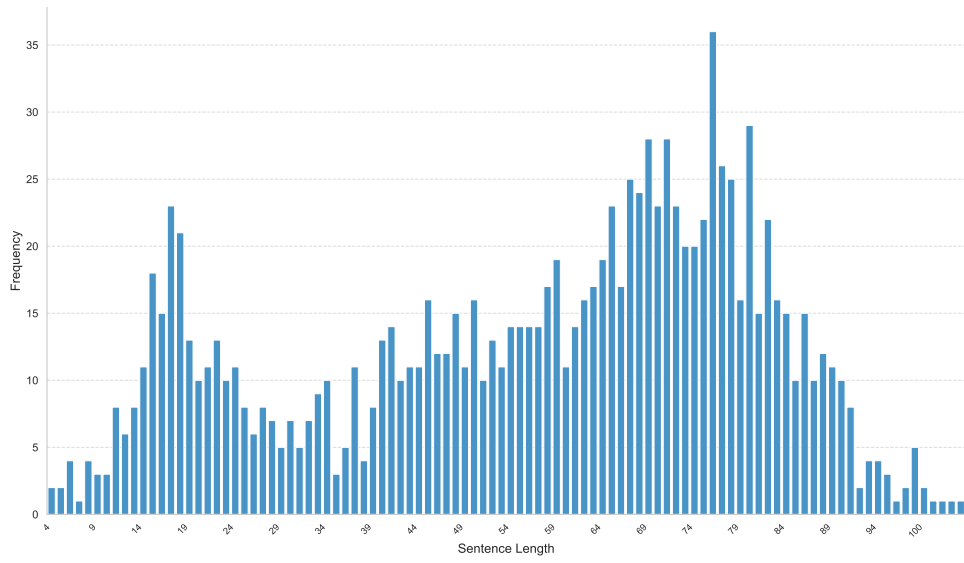


Figure 16: The software description length distribution in SRDD.

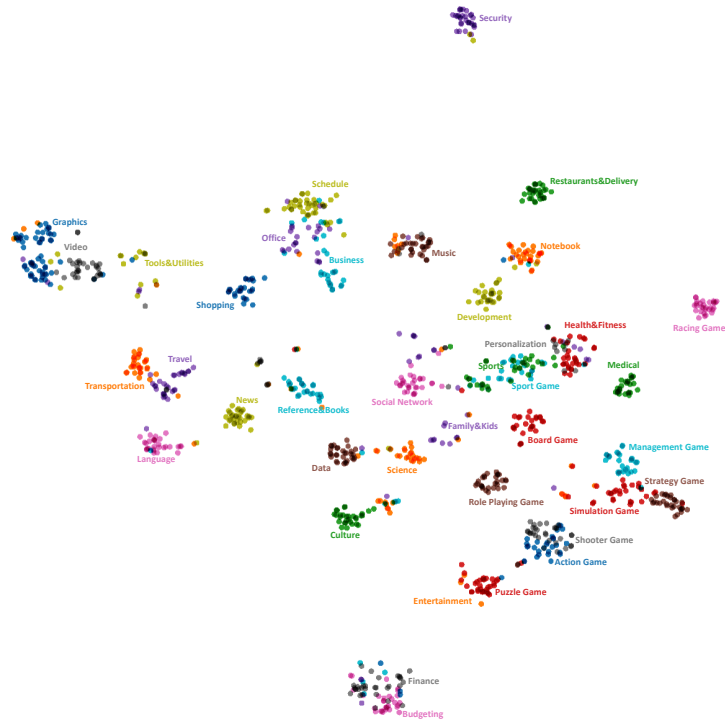


Figure 17: The t-SNE visualization of the SRDD task embeddings via the ada embedding model.

CommonGen Example

Concept set:

["oxidation", "cup", "mower", "space", "window", "arch", "pineapple", "spatula", "smoke", "tie", "gun", "insert", "boat", "use", "tomato", "chest", "vest", "bartender", "move", "microphone", "axis"]

Coherent sentence:

In order to achieve success, one must be persistent in their efforts, just like a mower persistently cuts through the grass. It is important to keep pushing forward, even in the face of adversity, just like the smoke that rises even when the fire is trying to fade. Encouraging others to do the same can lead to a more positive and productive environment. It takes a daring attitude to step out of one's comfort zone and take risks, just like a boat daringly sailing into the vast space of the ocean. By being persistent, encouraging, and daring, one can overcome obstacles and achieve their goals, just like a pineapple growing steadily on its plant, even when it faces harsh weather conditions. I draw inspiration from pushing the boundaries of traditional art forms and exploring new techniques. It is through this bold approach that I am able to create truly unique and inspiring pieces. In my latest piece, I incorporated elements of oxidation to create a stunning visual representation of change and transformation. The image of a cup, delicately balanced on the edge of a window, symbolizes the delicate balance of life. I used a mower to carve intricate patterns into the grass, turning the lawn into a work of art. The vast expanse of space serves as a backdrop for my imagination to run wild, unrestricted by earthly limitations. I adorned the archway with vibrant pineapple sculptures, adding a touch of tropical flair to the serene surroundings. A spatula became my tool of choice as I manipulated smoke to create ethereal and ephemeral forms that danced in the air. As I carefully inserted a boat into the space, signifying a journey towards new horizons and discoveries, with each deliberate move, I guided the piece towards its final form, shaping it with the precision of a sculptor. The gun, once a symbol of destruction, was transformed into a statement of peace and reconciliation. The microphone, positioned at the axis of the composition, captured the subtle nuances and whispers of the artwork, amplifying its impact. In the end, I stood back and admired the culmination of my efforts, a testament to the power of daring creativity and fearless self-expression.

Figure 18: One example of CommonGen's concept-sets and the coherent sentences generated by MACNET.

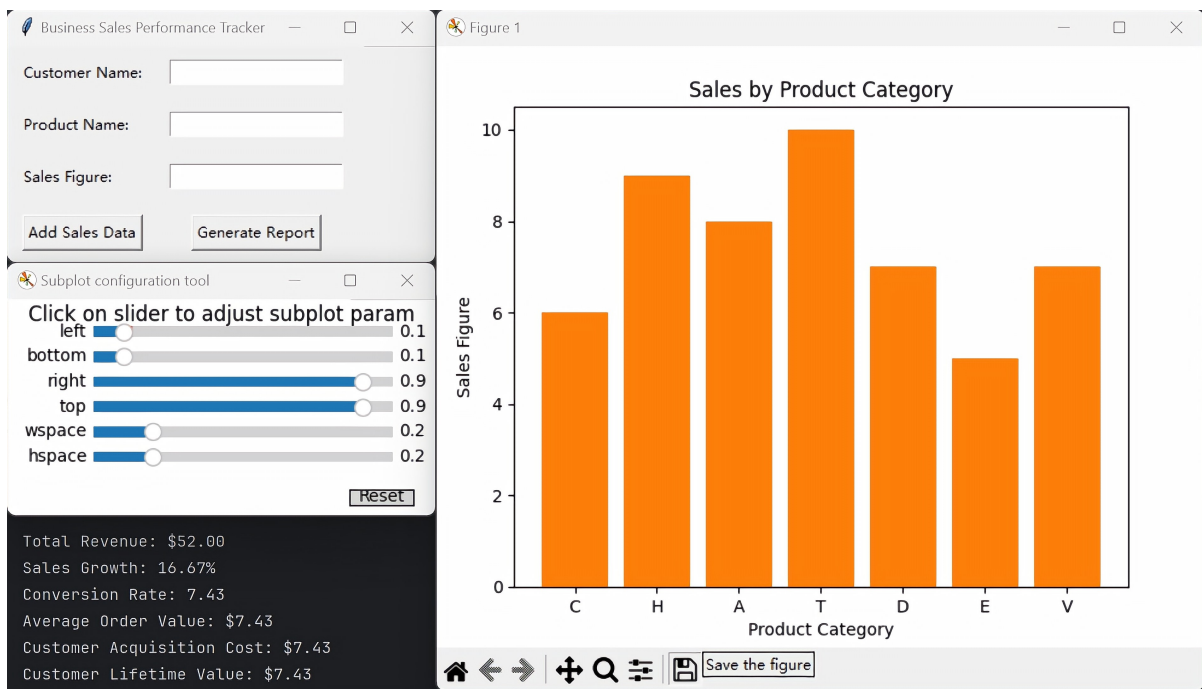


Figure 19: Demonstration of the "Business Sales Performance Tracker" software developed by MACNET-CHAIN.

Profiles



Instructor



Assistant



You are a highly skilled programmer, ready to embark on the development of a Business application. Your first notable characteristic is your exceptional ability to design effective algorithms and analyze their time and space complexity. With your keen understanding of algorithm design and analysis, you possess the expertise to create efficient and optimized solutions for complex problems. Your meticulous approach ensures that the algorithms you design are not only effective but also scalable, taking into consideration the time and space constraints of the application. In addition to your algorithmic prowess, you possess a second characteristic that sets you apart – software training and education. You have a natural talent for imparting knowledge and providing software training to both users and team members. Your ability to break down complex concepts into easily understandable modules allows you to effectively educate others on the intricacies of the software. Whether it's conducting workshops, creating comprehensive training materials, or offering one-on-one guidance, you excel at empowering individuals to confidently navigate the application. Lastly, your expertise extends to the formulation and execution of software version control strategies. You understand the importance of maintaining a well-organized and controlled development environment. With your meticulous attention to detail, you ensure that the software version control strategy is carefully crafted to facilitate seamless collaboration among team members. You implement best practices, such as branching and merging, to ensure the smooth integration of code changes. Your ability to navigate complex version control systems allows you to effectively manage multiple versions of the software, ensuring stability and reliability throughout the development process. As a programmer poised to develop a Business application, your exceptional skills in algorithm design and analysis, software training and education, and software version control strategy make you an invaluable asset. Your ability to create efficient algorithms, educate others on software usage, and maintain a controlled development environment sets the stage for a successful and impactful application.



As a programmer poised to develop a Business application, you possess a remarkable proficiency in multiple programming languages. Your mastery of languages like Python, Java, C++, JavaScript, and more, allows you to effortlessly navigate through various coding frameworks and libraries, ensuring a seamless development process. Beyond your programming language expertise, you have also honed your skills in leadership development within software teams. Your ability to cultivate and develop leadership qualities among your team members is commendable. You inspire and motivate your colleagues, fostering a collaborative and efficient work environment. Your guidance and mentorship empower your team to deliver high-quality software solutions, meeting and exceeding client expectations. Furthermore, you excel in providing software user training and support. Understanding the importance of user-friendly applications, you invest time and effort in ensuring that end-users are well-equipped to utilize the software effectively. Your comprehensive training sessions and ongoing support enable users to navigate the application effortlessly, maximizing its potential and enhancing their productivity. In summary, as a programmer poised to develop a Business application, your mastery of multiple programming languages, leadership development skills, and dedication to software user training make you an invaluable asset. Your ability to seamlessly navigate through coding languages, cultivate leadership qualities, and provide exceptional user support ensures the successful development and implementation of business applications.

Figure 20: Assign profiles to agents.

Agent Interaction in SRDD



Instructor



Assistant



We have decided to complete the task through a executable software with multiple files implemented via Python. As the Programmer, to satisfy the new user's demands, you should write one or multiple files and make sure that every detail of the architecture is, in the end, implemented as code. The software should be equipped with graphical user interface (GUI) so that user can visually and graphically use it; so you must choose a GUI framework (e.g., in Python, you can implement GUI via tkinter, Pygame, Flexx, PyGUI, etc.).

Think step by step and reason yourself to the right decisions to make sure we get it right.

You will first lay out the names of the core classes, functions, methods that will be necessary, as well as a quick comment on their purpose.



main.py

```
import tkinter as tk
from tkinter import messagebox
from sales_data import SalesData
from sales_report import SaleReport
class SalesPerformanceTracker:
    def __init__(self, root):
        self.root = root
        self.root.title("BusinessSalesPerformanceTracker")
        self.sales_data = SalesData()
        self.create_widgets()
    def create_widgets(self):
        # Create and configure input fields
        self.customer_name_label = tk.Label(self.root, text="CustomerName:")
        self.customer_name_label.grid(row=0, column=0, padx=10, pady=10, sticky=tk.W)
        self.customer_name_entry = tk.Entry(self.root)
        self.customer_name_entry.grid(row=0, column=1, padx=10, pady=10)
        self.product_name_label = tk.Label(self.root, text="ProductName:")
        self.product_name_label.grid(row=1, column=0, padx=10, pady=10, sticky=tk.W)
        self.product_name_entry = tk.Entry(self.root)
        self.product_name_entry.grid(row=1, column=1, padx=10, pady=10)
        self.sales_figure_label = tk.Label(self.root, text="SalesFigure:")
        self.sales_figure_label.grid(row=2, column=0, padx=10, pady=10, sticky=tk.W)
        self.sales_figure_entry = tk.Entry(self.root)
        self.sales_figure_entry.grid(row=2, column=1, padx=10, pady=10)
    def add_sales_data(self):
        customer_name = self.customer_name_entry.get()
        product_name = self.product_name_entry.get()
        sales_figure = self.sales_figure_entry.get()
        if customer_name and product_name and sales_figure:
            self.sales_data.add_data(customer_name, product_name, float(sales_figure))
            messagebox.showinfo("Success", "Salesdataaddedsuccessfully!")
        else:
            ...
```

Figure 21: Agent interaction, part 1.



Assistant

main.py(continued)

```

...

    else:
        messagebox.showerror("Error", "Pleasefillinallfields.")
        self.clear_input_fields()
    def generate_report(self):
        report = SalesReport(self.sales_data)
        report.generate()
    def clear_input_fields(self):
        self.customer_name_entry.delete(0, tk.END)
        self.product_name_entry.delete(0, tk.END)
        self.sales_figure_entry.delete(0, tk.END)
if __name__ == "__main__":
    root = tk.Tk()
    app = SalesPerformanceTracker(root)
    root.mainloop()

```

sales_data.py

```

class SalesData:
    def __init__(self):
        self.data = []
    def add_data(self, customer_name, product_name, sales_figure):
        self.data.append({
            "customer_name": customer_name,
            "product_name": product_name,
            "sales_figure": sales_figure
        })
    def get_total_revenue(self):
        return sum(data["sales_figure"] for data in self.data)
    def get_sales_growth(self):
        if len(self.data) < 2:
            return 0
        first_sales = self.data[0]["sales_figure"]
        last_sales = self.data[-1]["sales_figure"]
        return (last_sales - first_sales) / first_sales * 100
    def get_conversion_rate(self):
        if len(self.data) == 0:
            return 0
        total_sales = sum(data["sales_figure"] for data in self.data)
        total_customers = len(self.data)
        return total_sales / total_customers
    def get_average_order_value(self):
        if len(self.data) == 0:
            return 0
        total_sales = sum(data["sales_figure"] for data in self.data)
        total_orders = len(self.data)
        return total_sales / total_orders
    def get_customer_acquisition_cost(self):
        if len(self.data) == 0:
            return 0
        total_sales = sum(data["sales_figure"] for data in self.data)
        total_customers = len(self.data)
        return total_sales / total_customers
    def get_customer_lifetime_value(self):

```

...

Figure 22: Agent interaction, part 2.



Assistant

sales_data.py(continued)

```

...

def get_customer_lifetime_value(self):
    if len(self.data) == 0:
        return 0
    total_sales = sum(data["sales_figure"] for data in self.data)
    total_customers = len(self.data)
    return total_sales / total_customers

def get_sales_by_product_category(self):
    sales_by_category = {}
    for data in self.data:
        product_name = data["product_name"]
        sales_figure = data["sales_figure"]
        if product_name in sales_by_category:
            sales_by_category[product_name] += sales_figure
        else:
            sales_by_category[product_name] = sales_figure
    return sales_by_category

```

sales_report.py.py

```

import matplotlib.pyplot as plt
class SalesReport:
    def __init__(self, sales_data):
        self.sales_data = sales_data
    def generate(self):
        pass
    def generate_summary_report(self):
        total_revenue = self.sales_data.get_total_revenue()
        sales_growth = self.sales_data.get_sales_growth()
        conversion_rate = self.sales_data.get_conversion_rate()
        average_order_value = self.sales_data.get_average_order_value()
        customer_acquisition_cost = self.sales_data.get_customer_acquisition_cost()
        customer_lifetime_value = self.sales_data.get_customer_lifetime_value()
        report = f"SummaryReport:\n\n"
        report += f"TotalRevenue:${total_revenue:.2f}\n"
        report += f"SalesGrowth:{sales_growth:.2f}%\n"
        report += f"ConversionRate:{conversion_rate:.2f}\n"
        report += f"AverageOrderValue:${average_order_value:.2f}\n"
        report += f"CustomerAcquisitionCost:${customer_acquisition_cost:.2f}\n"
        report += f"CustomerLifetimeValue:${customer_lifetime_value:.2f}\n"
        print(report)
    def generate_sales_by_product_category_report(self):
        sales_by_category = self.sales_data.get_sales_by_product_category()
        categories = list(sales_by_category.keys())
        sales_figures = list(sales_by_category.values())
        plt.bar(categories, sales_figures)
        plt.xlabel("ProductCategory")
        plt.ylabel("SalesFigure")
        plt.title("SalesbyProductCategory")
        plt.show()

```



The main.py file contains the main application class `SalesPerformanceTracker`, which handles the GUI and user interactions. It imports the `SalesData` class from `sales_data.py` and the `SalesReport` class from `sales_report.py`. The `add_sales_data` method is responsible for...

Figure 23: Agent interaction, part 3.

Agent Interaction in SRDD



Instructor



Assistant



Unimplemented File

```
sales_report.py
```

In our software, each file must strictly follow a markdown code block format, where the following tokens must be replaced such that "FILENAME" is the lowercase file name including the file extension, "LANGUAGE" in the programming language, "DOCSTRING" is a string literal specified in source code that is used to document a specific segment of code, and "CODE" is the original code:

```
FILENAME
““ LANGUAGE
””
DOCSTRING
””
CODE
““
```

As the Programmer, to satisfy the complete function of our developed software, you have to implement all methods in the `sales_report.py` file which contains an unimplemented class. Now, implement all methods of the `sales_report.py` and all other codes needed, then output the fully implemented codes, strictly following the required format.



sales_report.py

```
...
def __init__(self, sales_data):
    self.sales_data = sales_data
def generate(self):
    self.generate_summary_report()
    self.generate_sales_by_product_category_report()
def generate_summary_report(self):
...

```

Code Diff on sales_report.py

```
def generate(self):
-    pass
+    self.generate_summary_report()
+    self.generate_sales_by_product_category_report()
```

Figure 24: Agent interaction, part 4.



Instructor



Assistant



{Codes of current version}
<Comment>

The highest priority comment is that the `get_customer_lifetime_value` method in the `SalesData` class is currently returning the same value as the `get_customer_acquisition_cost` method. This is incorrect as the customer lifetime value should be calculated differently.

To fix this, you need to update the logic in the `get_customer_lifetime_value` method to calculate the average sales per customer over the lifetime of the business. You can do this by dividing the total revenue by the total number of customers.

Here's the revised code for the `get_customer_lifetime_value` method:

sales_data.py

```
def get_customer_lifetime_value(self):
    if len(self.data) == 0:
        return 0
    total_sales = sum(data["sales_figure"] for data in self.data)
    total_customers = len(set(data["customer_name"] for data in self.data))
    return total_sales / total_customers
```

Please make this change and ensure that the `get_customer_lifetime_value` method returns the correct value.



sales_data.py

```
...
def get_customer_acquisition_cost(self):
    if len(self.data) == 0:
        return 0
    total_sales = sum(data["sales_figure"] for data in self.data)
    total_customers = len(self.data)
    return total_sales / total_customers
def get_customer_lifetime_value(self):
    if len(self.data) == 0:
        return 0
    total_sales = sum(data["sales_figure"] for data in self.data)
    total_customers = len(set(data["customer_name"] for data in self.data))
    return total_sales / total_customers
...
```

Code Diff on sales_data.py

```
def get_customer_lifetime_value(self):
    if len(self.data) == 0:
        return 0
    total_sales = sum(data["sales_figure"] for data in self.data)
-   total_customers = len(self.data)
+   total_customers = len(set(data["customer_name"] for data in self.data))
```

Figure 25: Agent interaction, part 5. To save space, we have omitted the specific code with {Codes of current version}.



Instructor



Assistant



{Codes of current version}

The highest priority comment is that the GUI elements of buttons for "Add Sales Data" and "Generate Report" are not implemented in the `SalesPerformanceTracker` class in `main.py`. To make the software user-friendly and provide intuitive controls for shape manipulation, the GUI elements need to be created.

To fix this, you need to implement the GUI layout and controls in the `SalesPerformanceTracker` class.

For example:

main.py

```
self.add_button = tk.Button()
self.add_button.grid()
self.generate_report_button = tk.Button()
self.generate_report_button.grid()
```

Make sure to replace the placeholder code with the actual implementation of the GUI elements and their corresponding event bindings and callback functions. Once these changes are made, the software will have a user-friendly interface with intuitive controls.



main.py

```
...
self.sales_figure_entry = tk.Entry(self.root)
self.sales_figure_entry.grid(row=2, column=1, padx=10, pady=10)
# Create buttons
self.add_button = tk.Button(self.root, text="AddSalesData", command=self.
    add_sales_data)
self.add_button.grid(row=3, column=0, padx=10, pady=10)
self.generate_report_button = tk.Button(self.root, text="GenerateReport", command=
    self.generate_report)
self.generate_report_button.grid(row=3, column=1, padx=10, pady=10)
def add_sales_data(self):
    customer_name = self.customer_name_entry.get()
    product_name = self.product_name_entry.get()
...

```

Code Diff on main.py

```
self.sales_figure_entry = tk.Entry(self.root)
self.sales_figure_entry.grid(row=2, column=1, padx=10, pady=10)
+ # Create buttons
+ self.add_button = tk.Button(self.root, text="AddSalesData", command=self.
    add_sales_data)
+ self.add_button.grid(row=3, column=0, padx=10, pady=10)
+ self.generate_report_button = tk.Button(self.root, text="GenerateReport", command=
    self.generate_report)
+ self.generate_report_button.grid(row=3, column=1, padx=10, pady=10)
```

Figure 26: Agent interaction, part 6. To save space, we have omitted the specific code with {Codes of current version}.

Agent Interaction in SRDD



Instructor



Assistant



Test Reports of Source Codes:

Test Error on sales_report.py

```
Traceback (most recent call last):
  File "sales_report.py", line 6, in <module>
    import matplotlib.pyplot as plt
ModuleNotFoundError: No module named 'matplotlib'
```

Error Summary of Test Reports: "nothing need to do on codes, install the missing module"

Note that each file must strictly follow Now, use the format exemplified above and modify the problematic codes based on the error summary. Output the codes that you fixed based on the test reported and corresponding explanations (strictly follow the format defined above, including FILENAME, LANGUAGE, DOCSTRING and CODE; incomplete "TODO" codes are strictly prohibited). If no bugs are reported, please return only one line like "<INFO> Finished".



<Action> Install matplotlib package

```
> pip3 install matplotlib
Collecting matplotlib
.....
Downloading matplotlib-3.8.2-cp39-cp39-macosx_11_0_arm64.whl (7.5 MB)
----- 7.5/7.5 MB 7.6 MB/s eta 0:00:00
Installing collected packages: matplotlib
Successfully installed matplotlib-3.8.2
```

sales_report.py

```
import matplotlib.pyplot as plt
class SalesReport:
    def __init__(self, sales_data):
        self.sales_data = sales_data
...
```

Figure 27: Agent interaction, part 7.

Agent Interaction in SRDD



Instructor



Assistant



Test Error on main.py

```
Test Reports of Source Codes:
Traceback (most recent call last):
  File "main.py", line 9, in <module>
    from sales_report import SaleReport
ImportError: cannot import name 'SaleReport' from 'sales_report' (sales_report.py)
```

Error Summary of Test Reports:

Based on the test report, the bug causing the problem is an import error in the main.py file. The error message states that it cannot import the name `SaleReport` from the `sales_report` module. This suggests that there is an issue with the import statement in the main.py file.

To fix this bug, you need to import the correct name from the `sales_report` module.

Note that each file must strictly follow Now, use the format exemplified above and modify the problematic codes based on the error summary. Output the codes that you fixed based on the test reported and corresponding explanations (strictly follow the format defined above, including FILENAME, LANGUAGE, DOCSTRING and CODE; incomplete "TODO" codes are strictly prohibited). If no bugs are reported, please return only one line like "<INFO> Finished".



main.py

```
'''
Business Sales Performance Tracker
Author: \textsc{MacNet-Chain}
'''
import tkinter as tk
from tkinter import messagebox
from sales_data import SalesData
from sales_report import SalesReport
class SalesPerformanceTracker:
    def __init__(self, root):
        self.root = root
...
'''
```

Code Diff on main.py

```
from sales_data import SalesData
- from sales_report import SaleReport
+ from sales_report import SalesReport
```

Figure 28: Agent interaction, part 8.

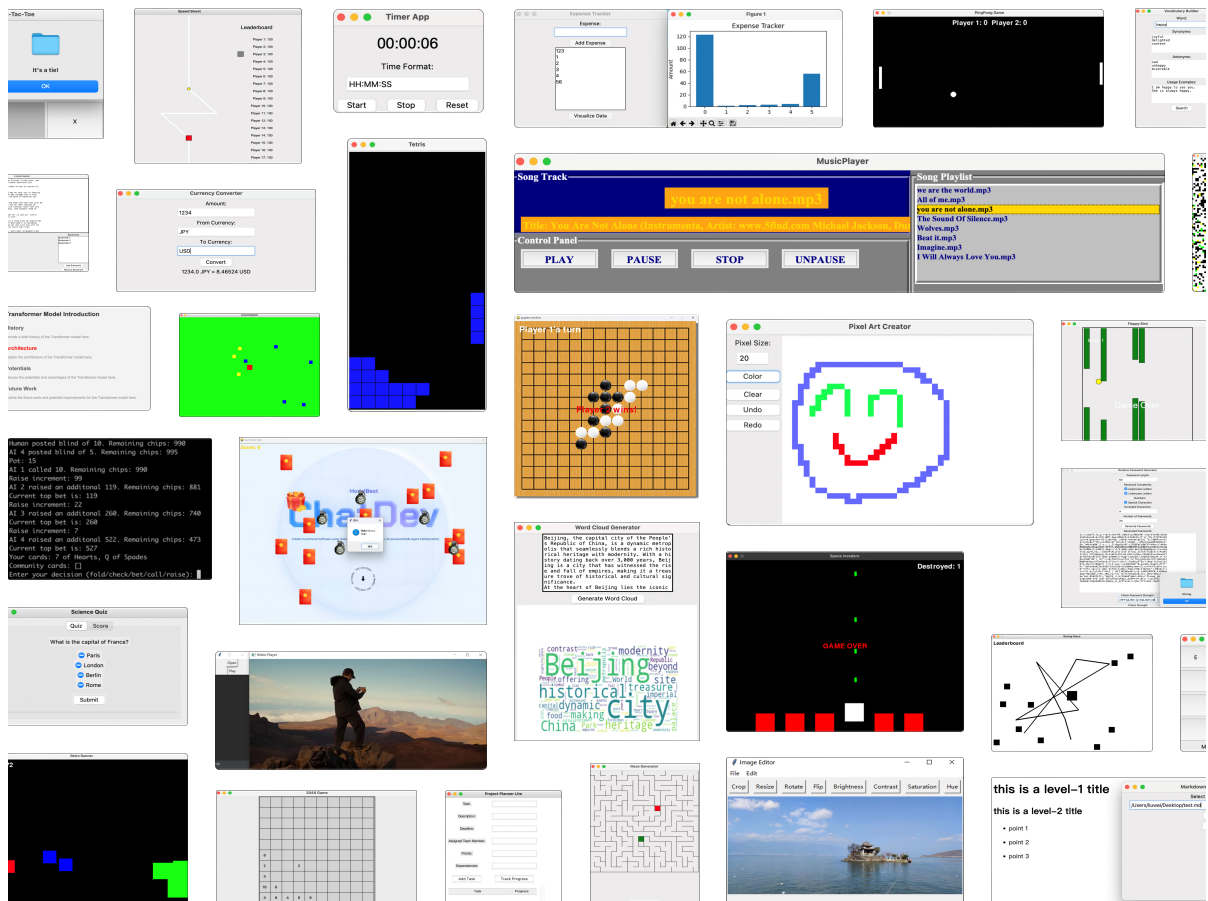


Figure 29: The software repository crafted by MACNET-CHAIN encompasses a diverse array of software categories, including but not limited to the game category and tool category. Each category contains a range of applications, each uniquely designed to meet specific user requirements and functionalities. The game category includes a variety of games developed using MACNET-CHAIN, ranging from simple puzzle games to more complex strategy and simulation games. These games are designed not only for entertainment but also to demonstrate the capabilities of MACNET-CHAIN in handling intricate logic, graphics, and user interaction. The tool category comprises various utility and productivity tools. Examples might include applications for data analysis, task management, or content creation. These tools are tailored to enhance productivity and efficiency, showcasing MACNET-CHAIN’s ability to create software that addresses practical, everyday needs. In addition to these categories, the MACNET-CHAIN-created software warehouse likely includes many other types of software, each illustrating the versatility and breadth of applications that can be developed using this advanced development platform.