Multi-Agent LLM Collaboration System to Code Generation

Haosheng Tan

January 2025

1 Introduction

Automatic code generation is a sub-task of automatic software development, which requires generating various formats of executable files with algorithms to fulfill certain functions according to the program specification. With the development of Natural Language Processing(NLP), code generation has been shifting from using formal logic and heuristics to automate the refinement of specifications[1] to treating code as another 'language' and embedding code generation into Large Language Models(LLMs). Conventional Software development follows the waterfall framework of Design, Coding, Code Review, and Testing procedures before achieving the requirement, and programmers are supposed to generate or modify code to meet specific requirements or fix potential bugs, which are also the expected goals of automatic coding. Equipped with the power of natural language understanding, reasoning, and tool using, LLMs are able to take the place of programming agents for code generation and bug fixing.

Large language models have been the most representative type of models in Natural Language Processing(NLP) in recent years and have exhibited remarkable proficiency across a wide range of domains. Conversational and chat-based LLMs has yielded a new formation of problem solving provided by human input. Instead of relying on human interleave conversation, several work has explored utilizing the collaborations between LLMs to achieve several goals[2, 3, 4, 5, 6, 7] including software development[2, 7], game playing[5, 2], and film making[6]. This system is so called multi-agent system(MAS) where agent is an intelligent entity that could take sequential actions of using tools, reasoning, and interacting with other entities to generate responses with user needs[8]. MAS shows superior advancement through communication and interaction among agents in the field of software development since software development could be divided into subtasks and accomplished by assigning subtasks to different members of the team. Typically, a MAS for software development involves procedure like role recruitment, collaborative discussion, execution and evaluation and through multiple iterations of these process, a final solution is given[2]. These systems demonstrate greater meticulous management and less hallucinations on specific requirement compared to single agent solutions and experiments [9, 10, 11, 12] shows that role-playing and collaborative communication significantly improve agent's performance by letting 'expert' agent specializes in a specific function. Conformity behavior, which allows an individual to adjust its behavior to align with the norms or goals of a group is also observed in the multi-agent interaction[2] and

cross-examination through communication could effectively reduce the hallucination, which shows a great potential in efficient large-scale tasks.

2 Challenges in multi-agent system to code generation

2.1 Hallucinations and Inherent Randomness

While MAS outperforms single-agent solutions on software development, it also shows some limitations. Firstly, inherent randomness and hallucination of the LLMs incur instability of MAS and pose an adverse impact on task completeness. LLMs show implicit randomness in response generation given the identical question[13, 2], and this feature ensures the diversity of the solutions. However, such randomness may lead to unstable solutions on the same task or even cause unalignment to the user's requirement. Although a temperature parameter is provided to adjust the randomness of an LLM, each software produced may vary between different runs in a low temperature configuration[13]. This indicates that MAS is suitable for open and creative software development where certain variations are acceptable. Besides, hallucination is a main problem of LLM ungrounding code generation. Many LLMs generate syntactically correct but non-executable code due to hallucinations, such as lack of dependencies, incomplete functions and potential bugs. Also, user-interfaces designed for human perception are not suitable for agent programmer where redundant visual details harass LLMs' generation as LLMs are sensitive to overall inputs[14]. Although cross-examination provided by other agents during decision-making could alleviate hallucinations, it is still a major issue especially in complex logic and syntax-heavy tasks[15] and lack of proper code version control or run-time evaluation process. Moreover, cross-agent validation also increases the communication cost and slows down the overall development, which shows a trade-off between software efficacy and efficiency.

2.2 Poor Executable Solutions and Lack of Robustness

Secondly, the generated solution is not grounding, showing weak alignment to human perspectives. Agent programmer may not generate code in human logic[13] and designer doesn't generate positive UI in users perspective occasionally[16]. One of the reason is unclear specification in the requirement[17], which leads to misunderstanding of the agent. Other reasons such as inter-agent misalignment caused by inefficient communication and conflict behaviors, and the limited modality of agents as well as redundant noise of inputs also contribute to the mistakes. The majority of modern agents are based on text-based LLM and interact with the environment described by text, while tasks like UI design depend largely on vision perception. Incorporating Vision-Language Models (VLM) and assigning different LLM backbone to agents corresponding to their tasks is a promising solution. Besides, the length of token may lead to code incompleteness in complicated software development[13], which has a catastrophic impact on software quality. And the increase in the number of conversation rounds also leads to a quadratic increase in cost as most of the dialogue leverages memory of the historical conversation. This also indicates a trade-off between solution quality and computation and memory budget. Insufficiently general solutions and unable to address

issues via feedback are the common cases of code generation failures, which shows a strong demand of expert knowledge in coding and indicates the need of fine-tuning.

2.3 Coordination Complexity and Error Propagation

Thirdly, inability to deal with coordination complexity and the instability within the framework is a common problem to be solved for MAS. Inter-framework errors such as role flipping and infinite error messages during communication between agents are widely witnessed in MAS[7, 13] and are typically caused by loopholes in the communication protocol or agents' violation of the rules. In addition, Error propagation is also a major problem either at system level or agent level for single-team agent systems working on tasks requiring completion of serial sub-tasks. Simple errors, such as a syntax error, can propagate at agent level and could not be recovered during auto-regressive code generation. And system level errors generally happen when one of the atomic tasks fails to produce an expected or a suboptimal solution is produced, especially in the framework designing phase. This problem is caused by the flow development framework and lack of self-correction mechanism within the system. Incorporating novel development models, such as multi-team agent collaboration, or introducing self-correction mechanisms may alleviate it.

3 Related Works

3.1 Autonomous Agents

Autonomous agent that could solve various complex tasks without further human intervention arises since LLMs-based agents show its strong in context learning and reasoning ability. To equip the agent with tool-use ability, Tool learning[18] aims to unleash the power of LLMs to effectively interact with certain APIs to solve various complex tasks. ToolFormer[19] and ToolLLM[18] propose a self-supervised learning method and an overall framework for data construction, supervised fine-tuning and evaluation respectively, to enable LLMs with flexible tool usage. Since strong reasoning ability is integrated in LLMs, React[20] and Reflextion[21] equip MAS with chain-of-thought(CoT)[22, 23] reasoning for action plans generations and make further reasoning or actions based on the observation. In addition, ReAct proposes a thought-action-observation formation that generates a more grounded solution using external knowledge, while CoT suffers from hallucinated thoughts due to reliance on self-knowledge and lack of environment feedback. Reflexion introduces Chian-of-thought prompts for reinforcement learning in MAS where feedback is given verbally, and short-term memory for fine-grained details as well as long-term memory for distilled experience is provided when an actor generates output. But this two framework also suffer from the limit of token length given large action space and trajectory history. Following the framework of ReAct, InterCode[24] propose a interactive virtual environment defined by Docker container for agent programmer and regards feedback as observation after taken action at current state. Yang et al.[14] reveal the adverse impacts of redundant information of interface for agent programmer and propose a informative and concise agent interface. These work show the great potential of agent in automatic code generation and the significance of interactive environment as well as precise feedback.

Several works also discover the way of using cooperation among agents for software development. CAMEL[7] defines a straightforward bi-agent interactive framework which involves two agents and accomplish programming tasks by communication. This framework assigns roles for different agents during the problem-solving process and introduces conversational solution that requires less human intervention. Also, it leverages a symmetric discoursed prompt format to ensure effective dialogue between roles. However, the instructor and assistant mode in this paradigm doesn't introduce reviewer for cross-evaluation and run-time executable feedback, which causes great hallucinations and low executable solutions and role flipping, infinite error message and flake reply are fatal for tasks completeness. Qian et al.[13] extends this paradigm by treating it as an atomic conversation and propose the chain of chats for different sub-tasks in software development. This framework incorporates different roles for different processes and it follows the waterfall architecture in software development, which makes it flexible to inject user's requirement into intermediate processes. In addition, role-switching for more specific instruction and it propose self-reflection, a technique to help agent better summarize historical communication in case of information loss. Nevertheless, it fails to handle the inherent randomness in the generated output even with low temperature setting and the length of token hinder the complete code generation in complicated software development.

In general, Autonomous Agents prosper from tool-use ability and conversational problem-solving interaction among agents. Chian-of-thought prompts and in-context learning ability of LLMs equips agents with strong reasoning power and the framework of action-thought-observation from ReAct[20] has been proven effective for injecting external knowledge. However, hallucinations and inherent randomness remain problems even when a low temperature parameter setting is given, and the limit of token length is a major problem for code generation. Moreover, problems like role-flipping and infinite error messages are potential loopholes of MAS, which require a more robust and reliable communication protocol and cooperation framework design.

3.2 LLM-Based Multi-Agent Frameworks

LLM-based multi-agent collaboration system have achieved astonishing performance in both industry and academia and flexibility and human-in-the-loop framework is the emerging trend of MAS. Various frameworks[13, 16, 25, 2, 5, 6] are proposed for MAS and be applied to different applications such as programming, social simulation and artistic creation. Among them, Park et al[5] proposes a framework and perform society simulation in a sandbox environment with 25 agents. To address long-term plan and memory retrieval, Generative Agents present a similar structure proposed by Shinn et al.[21] where observation is stored in memory database and relevant memory retrieval is performed if necessary. In addition, high-level reflection and long-term plan are generated from the retrieved memory, which are also stored in the memory database episodically. Yet, the information retriever is not robust and the computational cost for the overall framework tremendous, which makes it unfeasible for a long-time simulation. Inspired by CAMEL[7], ChatDev[13] develops a chat-chain framework that simulates the waterfall development architecture in programming. The atomic conversation structure is extended from CAMEL for communication between two agents and memory stream as well as self-reflection are introduced to reduce information loss. This framework formalize the MAS in software development with a well-defined framework and mitigate hallucinations with role-switching and self-reflection mechanism. Yet, it still suffers from inherent randomness and token length limit for code generation. Also, as waterfall framework divide

software development into sequential sub-tasks, error propagation is a huge problem as suboptimal intermediate process may lead to inferior result eventually. To address this problem, Du et al.[16] propose a cross-team collaboration framework which introduce cross-team evaluation and aggregation to generate optimal solution. During key phases of software development, solutions presented from various team would be allocated and pruned to ensure high quality candidates which are then aggregated by combining the strengths of different candidates and eliminating the weaknesses. This framework significantly improves the code executability as mutual correction and enlightenment enhance the software quality, and the aggregation of teams' content shows better alignment to the user requirements. However, this framework increases the computation cost incurred by multi-teams and the greedy aggregation may filter out potential valuable insights. Also, giving unexplicit requirement lead to the most straightforward design instead of a more human acceptable way while complicated requirement result in suboptimal solution by overfitting certain requirements, which shows a challenge of MAS in software development.

While Qian et al.[13] and Du et al.[16] leverage waterfall architecture, Chen et al.[2] propose an iterative framework by recruiting different roles based on the current states. A 'manager' agent recruits agents with different roles and skills at the start of the iteration according to the current state and the overall goal. Then, a group of recruited experts make decision through communication and take corresponding actions. Evaluation of the overall goal and current state is carried out by the manager after all the actions are accomplished for this iteration and start the next iteration by recruiting again until the overall goal is achieved. This framework gives a high flexibility of role recruitment for various tasks but keeping consistency of the solutions is a huge challenge. Since the majority of MAS relies on dialogue, Hong et al[15] improve ChatDev by introducing chat formation as normal conversation is a information transmission with potential information loss. In this framework, strict formation is strictly imposed for conversation, and a subscription-publish mechanism to a public information pool is developed, which reduces the information loss caused by one-to-one conversation. In addition, real-time executable feedback is provided for software engineer such that self-correction is greatly improved. This framework indicates a great potential of imposing Standardized Operating Procedures(SOPs) into MAS in software development and interactive executability feedback is essential for software engineering.

In summary, multiple MAS frameworks has been developed for various applications and currently, software development is the most popular field. Cross-team collaboration and SOPs formation are introduced to alleviate the hallucinations significantly and information loss could be mitigated by leveraging shared information mechanism. Yet, the consistency of MAS solutions is still a problem and precise user requirement is indispensable. Also, the limitation of token length poses a great challenge for complicated software development.

4 Methodology

This section briefly summarizes the methodologies that might be investigated in this research project. The proposed project could focus on one of the technologies to address current problems or investigate combination for novel solutions.

4.1 Multi-team Agent Collaboration

As single team MAS suffer from error propagation and hallucinations caused by individuals, multi-team collaboration with cross-team evaluation is a promising solution. Single-team agent collaboration decomposes the overall task O into different phrases $\mathcal{P}$ where each phrase consists of several sub-task $\mathcal{T}$. A single team execution is denoted as

$$\mathcal{C} = \langle \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \cdots, \mathcal{P}_{|\mathcal{C}|} \rangle \tag{1}$$

$$\mathcal{P}_i = \left\langle \mathcal{T}_i^1, \mathcal{T}_i^2, \mathcal{T}_i^3, \cdots, \mathcal{T}_i^{|\mathcal{P}^i|} \right\rangle \tag{2}$$

And for each of the sub-task, solutions are derived from communication between *instructor* and *assistant*:

$$\mathcal{T}_i^j = \tau \left(D \left(\mathcal{I}_i^j, \mathcal{A}_i^j \right) \right) \tag{3}$$

where τ is the solution and $D\left(\mathcal{I}_i^j, \mathcal{A}_i^j\right)$ denotes the dialogue. If the phrases are executed sequentially, such as the waterfall framework proposed by ChatDev[13], then the tasks of $\mathcal{P}_i$ depend on the previous phrase solution $\mathcal{P}_{i-1}$ as shown in (4).

$$\hat{\tau} = \underset{\tau}{\operatorname{argmax}}\, P(\mathcal{P}_i | \mathcal{P}_{i-1}, O) \tag{4}$$

If the outcome of $\mathcal{P}_i$ is suboptimal, i.e. the design of the software doesn't meet the user requirement, final outcome of $\mathcal{P}_{|\mathcal{C}|}$ is not able to satisfy the requirement as previous error propagates and accumulates at the final task. Thus, Cross-Team Collaboration[16] is introduced and it sets a new formation of scaling for MAS. CTC orchestrates parallel execution of multiple single-team execution denoted as $S = \left\{\mathcal{C}^1, \mathcal{C}^2, \mathcal{C}^3, \cdots, \mathcal{C}^n\right\}$ with n teams with hyperparameters, such as diversity and the maximum number of tasks. Instead of running n execution parallelly, during key phases $\mathcal{P}^{\mathcal{K}}$, aggregation mechanism of different teams is performed to raise mutual awareness. Contents generated by each team is pruned and partitioned into different groups for hierarchical aggregation where contents in a partition are aggregated by combining their strengths and eliminating the drawbacks. This process could be executed in an iterative process as shown as (5) to (8).

$$S^t = \langle \mathcal{X}_1^t, \mathcal{X}_2^t, \mathcal{X}_3^t, \ldots, \mathcal{X}_m^t \rangle \tag{5}$$

$$G^t = \left\langle \mathcal{G}_1^t, \mathcal{G}_2^t, \mathcal{G}_3^t, \cdots, \mathcal{G}_{\frac{m}{k}}^t \right\rangle \tag{6}$$

$$R_t = \left\langle \mathcal{R}_1^t, \mathcal{R}_2^t, \mathcal{R}_3^t, \cdots, \mathcal{R}_{\frac{m}{k}}^t \right\rangle \tag{7}$$

$$\mathcal{X}_i^{t+1} = \mathcal{R}_i^t \tag{8}$$

where m is the number of contents after pruning and k is the size of partition. t denotes the current depth of the iteration of the aggregation. $\mathcal{X}$ is the generated content for each team and $\mathcal{G}$ denotes the partitioned group. $\mathcal{R}$ is the aggregation result which would join the next iteraction of aggregation until $|S_t| = 1$. Hierarchical aggregation effectively combining strengths of various outcome while

alleviating information overflow for a single agent to digest. As cross-team evaluation is considered during intermediate processes, suboptimal solution caused by individual hallucination could be fixed effectively. However, greedy pruning potentially filters out creative contents, which reduce diversity of the solution and this process requires gigantic computational cost and tremendous communication bandwidth. Applying content aggregation only at key phases apart from large content and introduce fine-grained pruning instead of greedy pruning is considerable.

On this basis, further works could scale the CTC framework with better architectures. Ensuring diversity is essential as solutions from different teams are supposed to cover the solution space as much as possible, which guarantees more optimal results. Moreover, injecting human feedback into the intermediate processes provides a correction mechanism as well as higher machine-human interaction for the user's unique requirements. In addition, a better content aggregation mechanism is required to enhance aggregation quality. Flexible pruning mechanism including fine-grained features as well as the global view is a promising improvement.

4.2 Multi-modality Agent System

Since most of the MAS are based on language-based agents, scenarios such as UI design and visualization requires the input of multi-modal input. Text-only agent may fail to effectively communicate the detailed properties of objects, such as spatial features. Thus, introducing flexible multi-modality agent is a potential to improve content quality. Current multi-modality models mainly combine vision and language using modality fusion mechanism, which are Vision Language Models(VLMs). Several VLMs[26, 27, 28, 29] relies on 'dual tower' architecture where ViT-based vision encoder generates vision embedding and BERT-based text encoder generate language embedding followed by a modality fusion mechanism, which are cross-attention or dot product. Other VLMs[30, 31, 32] treat vision as a foreign language and propose using multi-modal sentence that interleave text tokens and vision tokens as the input of LLM. These methods could be summarized as using ViT-based into vision encoder $\phi : \mathcal{V} \rightarrow \mathcal{O}$ where $\mathcal{V}$ is vision space and $\mathcal{O}$ denotes the embedding space. Additional adapter φ plays a role in aligning vision and language by projections $\varphi : \mathcal{O} \rightarrow \mathcal{T}$ where $\mathcal{T}$ is the language space. After that, cross-attention or multi-modal sentence could be adopted to fuse modality. LLaVA-plus[33] propose a 'skill repository' with a number of models specialized in certain tasks, such as image segmentation and image generation. Apart from dialogue between instructor and assistant, a skill-oriented dialogue is implemented between agent and skill models which are regarded as tools. This framework effectively equips VLMs with ability of tool-using but fail to address the hallucination and the problem of tool conflicts. Introducing additional modality is effective for improving overall performance but flexibility is strongly required for tool-based agents. In addition, more modality would increase computation cost especially for agents that continuously require input of the environment. Properly dealing with the balance between computation cost and content quality is not a trivial problem. Besides, since important and redundant information is consumed equivalent computation resources for LLMs and it may fail to filter out noise from significant signal, interface for VLMs in code generation should be properly designed to avoid information overflow.

Future directions lie in better structure incorporating tool use in multi-modal models either by imposing meticulously crafted prompts or introducing VLM 'specialist' on specific tasks. Interfaces

for multi-modal agents are highly related to VLMs' perception. Besides, assigning multi-modality agents in MAS is critical as redundancy of multi-modality agents leads to additional computation cost. A flexible framework embedding multi-modality agents needs to be developed.

4.3 Scalable and Adaptive Reinforcement Learning-based MAS

Since LLM-based Multi-agent systems have achieved impressive success in basic scenarios, advancing it to accomplish more complex tasks is a definite trend. Benefits brought by scaling are either in scaling the size of LLMs or scaling the number of collaborative agents of which the latter one doesn't rely on the advancement of LLMs and could be achieved swiftly. Therefore, scalability is indispensable in MAS and techniques to manage the communication among agents in scaling situations while maintaining the efficiency is fundamental. To tackle the quadratic increment of communication cost, sharing information mechanism is critical. Shared Information pool proposed by Hong et al.[15] effectively avoids additional communication and information could be directly derived through public information pool. With the publish and subscription mechanism, agents could easily get information from the information pool and share their conversation to avoid additional communication. Also, this method successfully decreases the information loss during conversation between agents which is a major reason of role flipping, infinite error message or flake reply. Nevertheless, information pool also incurs the problem of information overload and irrelevant information retrieval, especially in a scalable framework. Lack of priority management and duplicate information would also cause conflicts and hallucinations. Thus, further works could be optimizing the information retrieving mechanism by introducing management of the information pool, such as ranking information by its priority and content aggregation proposed by Du et al[12]. To address information overload of agents to irrelevant information, hierarchical and distributed information storage where each agents could be subscribed to a set of sub-pools is a promising solution.

Apart from efficient communication, adaptability is also required for MAS in various complicated scenarios. Modern MAS suffers from poor generalization due to task-oriented framework design and fails to adapt to various and unpredicted environments, such as household applications. Adaptability is a critical metric to evaluate the flexibility of a MAS and to improve the flexibility, Reinforcement Learning(RL) gains great potential. Multi-Agent Reinforcement Learning(MARL) enables agents to adjust actions and plans based on the feedback from the environment, which is an effective way in improving adaptability. Leveraging this technique in MAS may offer systems flexibility to dynamically adjust roles assignment and tasks distribution based on various task requirements. Moreover, applying Markov Decision Processes (MDP) for SOP effectiveness evaluation could render system ability to modify SOPs dynamically based on the past task performance, continuously improving overall efficiency and presents great adaptation. Besides,Reinforce Learning with Human Feedback(RLHF)[34] is also a method to avoid adverse behaviours and align agent's behaviours with human expectations since emergent hazardous behaviors might pose risks, especially when humans are involved in the collaboration or work process[2]. The challenge incurred by RL is the instability of training caused by suboptimal reward functions, and the requirement of large amount of samples. A deployable and efficient RL method to gain MAS flexibility and great adaptation is worth investigation.

References

[1] R. Balzer, "A 15 Year Perspective on Automatic Programming," Tech. Rep. 11, 1985. 1

[2] W. Chen, Y. Su, J. Zuo, C. Yang, C. Yuan, C.-M. Chan, H. Yu, Y. Lu, Y.-H. Hung, C. Qian, Y. Qin, X. Cong, R. Xie, Z. Liu, M. Sun, and J. Zhou, "AgentVerse: Facilitating Multi-Agent Collaboration and Exploring Emergent Behaviors," *arXiv preprint arXiv:2308.10848*, vol. 2, p. 6, 8 2023. [Online]. Available: http://arxiv.org/abs/2308.10848 1, 2.1, 3.2, 4.3

[3] L. Wang, J. Zhang, H. Yang, Z. Chen, J. Tang, Z. Zhang, X. Chen, Y. Lin, R. Song, W. X. Zhao, J. Xu, Z. Dou, J. Wang, and J.-R. Wen, "User Behavior Simulation with Large Language Model based Agents," *ACM Transactions on Information Systems*, vol. 43, pp. 1–37, 6 2023. [Online]. Available: http://arxiv.org/abs/2306.02552 1

[4] J. Wei, K. Shuster, A. Szlam, J. Weston, J. Urbanek, and M. Komeili, "Multi-Party Chat: Conversational Agents in Group Settings with Humans and Models," *arXiv preprint arXiv:2304.13835*, 4 2023. [Online]. Available: http://arxiv.org/abs/2304.13835 1

[5] J. S. Park, J. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative Agents: Interactive Simulacra of Human Behavior," in *UIST 2023 - Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. Association for Computing Machinery, Inc, 10 2023. 1, 3.2

[6] Z. Xu, L. Wang, J. Wang, Z. Li, S. Shi, X. Yang, Y. Wang, B. Hu, J. Yu, and M. Zhang, "FilmAgent: A Multi-Agent Framework for End-to-End Film Automation in Virtual 3D Spaces," *arXiv preprint arXiv:2501.12909*, 1 2025. [Online]. Available: http://arxiv.org/abs/2501.12909 1, 3.2

[7] G. Li, H. A. A. K. Hammoud, H. Itani, D. Khizbullin, and B. Ghanem, "CAMEL: Communicative Agents for "Mind" Exploration of Large Language Model Society," *Advances in Neural Information Processing Systems*, vol. 36, pp. 51 991–52 008, 3 2023. [Online]. Available: http://arxiv.org/abs/2303.17760 1, 2.3, 3.1, 3.2

[8] Y. Talebirad and A. Nadiri, "Multi-Agent Collaboration: Harnessing the Power of Intelligent LLM Agents," *arXiv preprint arXiv:2306.03314*, 6 2023. [Online]. Available: http://arxiv.org/abs/2306.03314 1

[9] L. Salewski, S. Alaniz, I. Rio-Torto, E. Schulz, and Z. Akata, "In-Context Impersonation Reveals Large Language Models' Strengths and Biases," *Advances in neural information processing systems*, vol. 36, pp. 72 044–72 057, 2023. [Online]. Available: https://chat.lmsys.org/?leaderboard 1

[10] T. Liang, Z. He, W. Jiao, X. Wang, Y. Wang, R. Wang, Y. Yang, S. Shi, and Z. Tu, "Encouraging Divergent Thinking in Large Language Models through Multi-Agent Debate," *arXiv preprint arXiv:2305.19118*, 5 2023. [Online]. Available: http://arxiv.org/abs/2305.19118 1

[11] R. Liu, R. Yang, C. Jia, G. Zhang, D. Zhou, A. M. Dai, D. Yang, and S. Vosoughi, "Training Socially Aligned Language Models on Simulated Social Interactions," *arXiv preprint arXiv:2305.16960*, 5 2023. [Online]. Available: http://arxiv.org/abs/2305.16960 1

[12] Y. Du, S. Li, A. Torralba, J. B. Tenenbaum, and I. Mordatch, "Improving Factuality and Reasoning in Language Models through Multiagent Debate," in *Forty-first International Conference on Machine Learning*, 2023. 1, 4.3

[13] C. Qian, W. Liu, H. Liu, N. Chen, Y. Dang, J. Li, C. Yang, W. Chen, Y. Su, X. Cong, J. Xu, D. Li, Z. Liu, and M. Sun, "ChatDev: Communicative Agents for Software Development," *arXiv preprint arXiv:2307.07924*, 7 2023. [Online]. Available: http://arxiv.org/abs/2307.07924 2.1, 2.2, 2.3, 3.1, 3.2, 4.1

[14] J. Yang, C. E. Jimenez, A. Wettig, K. Lieret, S. Yao, K. Narasimhan, and O. Press, "SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering," *Advances in Neural Information Processing Systems*, vol. 37, pp. 50 528–50 652, 5 2024. [Online]. Available: http://arxiv.org/abs/2405.15793 2.1, 3.1

[15] S. Hong, M. Zhuge, J. Chen, X. Zheng, Y. Cheng, C. Zhang, J. Wang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou, C. Ran, L. Xiao, C. Wu, and J. Schmidhuber, "MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework," *arXiv preprint arXiv:2308.00352*, vol. 3, p. 6, 8 2023. [Online]. Available: http://arxiv.org/abs/2308.00352 2.1, 3.2, 4.3

[16] Z. Du, C. Qian, W. Liu, Z. Xie, Y. Wang, Y. Dang, W. Chen, and C. Yang, "Multi-Agent Software Development through Cross-Team Collaboration," *arXiv preprint arXiv:2406.08979*, 6 2024. [Online]. Available: http://arxiv.org/abs/2406.08979 2.2, 3.2, 4.1

[17] M. Cemri, M. Z. Pan, S. Yang, L. A. Agrawal, B. Chopra, R. Tiwari, K. Keutzer, A. Parameswaran, D. Klein, K. Ramchandran, M. Zaharia, J. E. Gonzalez, and I. Stoica, "Why Do Multi-Agent LLM Systems Fail?" *arXiv preprint arXiv:2503.13657*, 3 2025. [Online]. Available: http://arxiv.org/abs/2503.13657 2.2

[18] Y. Qin, S. Liang, Y. Ye, K. Zhu, L. Yan, Y. Lu, Y. Lin, X. Cong, X. Tang, B. Qian, S. Zhao, L. Hong, R. Tian, R. Xie, J. Zhou, M. Gerstein, D. Li, Z. Liu, and M. Sun, "ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs," *arXiv preprint arXiv:2307.16789*, 7 2023. [Online]. Available: http://arxiv.org/abs/2307.16789 3.1

[19] T. Schick Jane Dwivedi-Yu Roberto Dessì and R. Raileanu Maria Lomeli Eric Hambro Luke Zettlemoyer Nicola Cancedda Thomas Scialom FAIR, "Toolformer: Language Models Can Teach Themselves to Use Tools," *Advances in Neural Information Processing Systems*, vol. 36, pp. 68 539–68 551, 2023. 3.1

[20] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing Reasoning and Acting in Language Models," in *International Conference on Learning Representations (ICLR)*, 10 2023. [Online]. Available: http://arxiv.org/abs/2210.03629 3.1

[21] N. Shinn, F. Cassano, E. Berman, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language Agents with Verbal Reinforcement Learning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 8634–8652, 3 2023. [Online]. Available: http://arxiv.org/abs/2303.11366 3.1, 3.2

[22] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi Quoc, V. Le, and D. Zhou, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models Chain-of-Thought Prompting," Tech. Rep. 3.1

[23] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-Consistency Improves Chain of Thought Reasoning in Language Models," *arXiv preprint arXiv:2203.11171*, 3 2022. [Online]. Available: http://arxiv.org/abs/2203.11171 3.1

[24] J. Yang, A. Prabhakar, K. Narasimhan, and S. Yao, "InterCode: Standardizing and Benchmarking Interactive Coding with Execution Feedback," *Advances in Neural Information Processing Systems*, vol. 36, pp. 23 826–23 854, 6 2023. [Online]. Available: http://arxiv.org/abs/2306.14898 3.1

[25] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, A. H. Awadallah, R. W. White, D. Burger, and C. Wang, "AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation," *arXiv preprint arXiv:2308.08155*, 8 2023. [Online]. Available: http://arxiv.org/abs/2308.08155 3.2

[26] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning Transferable Visual Models From Natural Language Supervision," in *International conference on machine learning*, 2 2021, pp. 8748–8763. [Online]. Available: http://arxiv.org/abs/2103.00020 4.2

[27] J. Li, R. R. Selvaraju, A. D. Gotmare, S. Joty, C. Xiong, and S. Hoi, "Align before Fuse: Vision and Language Representation Learning with Momentum Distillation," *Advances in neural information processing systems*, vol. 34, pp. 9694–9705, 7 2021. [Online]. Available: http://arxiv.org/abs/2107.07651 4.2

[28] J. Li, D. Li, C. Xiong, and S. Hoi, "BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation," in *International conference on machine learning*. PMLR, 1 2022, pp. 12 888–12 900. [Online]. Available: http://arxiv.org/abs/2201.12086 4.2

[29] J. Yu, Z. Wang, V. Vasudevan, L. Yeung, M. Seyedhosseini, and Y. Wu, "CoCa: Contrastive Captioners are Image-Text Foundation Models," *arXiv preprint arXiv:2205.01917*, 5 2022. [Online]. Available: http://arxiv.org/abs/2205.01917 4.2

[30] Y. Hao, H. Song, L. Dong, S. Huang, Z. Chi, W. Wang, S. Ma, and F. Wei, "Language Models are General-Purpose Interfaces," *arXiv preprint arXiv:2206.06336*, 6 2022. [Online]. Available: http://arxiv.org/abs/2206.06336 4.2

[31] S. Huang, L. Dong, W. Wang, Y. Hao, S. Singhal, S. Ma, T. Lv, L. Cui, O. Khan Mohammed, B. Patra, Q. Liu, K. Aggarwal Zewen Chi, J. Bjorck, V. Chaudhary, S. Som, X. Song, and F. Wei, "Language Is Not All You Need: Aligning Perception with Language Models," *Advances in Neural Information Processing Systems*, vol. 36, pp. 72 096–72 109, 2023. 4.2

[32] W. Wang, H. Bao, L. Dong, J. Bjorck, Z. Peng, Q. Liu, K. Aggarwal, O. K. Mohammed, S. Singhal, S. Som, and F. Wei, "Image as a Foreign Language: BEiT Pretraining for All Vision and Vision-Language Tasks," *arXiv preprint arXiv:2208.10442*, 8 2022. [Online]. Available: http://arxiv.org/abs/2208.10442 4.2

[33] S. Liu, H. Cheng, H. Liu, H. Zhang, F. Li, T. Ren, X. Zou, J. Yang, H. Su, J. Zhu, L. Zhang, J. Gao, and C. Li, "LLaVA-Plus: Learning to Use Tools for Creating Multimodal Agents," in *European Conference on Computer Vision*. Springer, 11 2023, pp. 126–142. [Online]. Available: http://arxiv.org/abs/2311.05437 4.2

[34] Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan, N. Joseph, S. Kadavath, J. Kernion, T. Conerly, S. El-Showk, N. Elhage, Z. Hatfield-Dodds, D. Hernandez, T. Hume, S. Johnston, S. Kravec, L. Lovitt, N. Nanda, C. Olsson, D. Amodei, T. Brown, J. Clark, S. McCandlish, C. Olah, B. Mann, and J. Kaplan, "Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback," *arXiv preprint arXiv:2204.05862*, 4 2022. [Online]. Available: http://arxiv.org/abs/2204.05862 4.3