A State-of-the-Art Survey on AI Systems for Multi-Task Productivity: From Large Language Models to Multi-Agent Systems

Your Name
Your Affiliation

***Abstract*—The rapid evolution of large language models (LLMs) has catalyzed the development of systems capable of performing multiple productive tasks. Early breakthroughs in language modeling paved the way for models to autonomously select and utilize external tools. Subsequent research focused on transforming these models into Large Action Models (LAMs) tailored for action execution. More recently, integrating LAMs within multi-agent systems (MASs) has emerged as a promising strategy for tackling complex, real-world tasks. This survey synthesizes historical progress, contemporary research, and future directions in the field—from LLMs using tools to MASs that achieve collaborative, autonomous task execution.**

***Index Terms*—LLMs, Toolformer, ToolLLM, Large Action Models, Multi-Agent Systems, Autonomous Agents, Collaborative AI**

I. Introduction

Large language models (LLMs) have redefined what is possible in artificial intelligence, demonstrating unprecedented capabilities in natural language understanding and generation. Despite their success, early LLMs were inherently limited to text-based outputs and struggled with tasks requiring real-world action. Recent work has demonstrated that LLMs can be extended to identify and call appropriate tools, effectively bridging the gap between language and action. This survey reviews the evolution from tool-using LLMs to specialized Large Action Models (LAMs), and ultimately to multi-agent systems that harness these capabilities for collaborative, productive task execution.

In the following sections, we outline:

- **How we got here:** A brief history of LLMs' transformation through tool usage.
- **Where we are now:** The emergence of LAMs and their integration into multi-agent frameworks.
- **Where we could go next:** Challenges faced by current systems and potential directions for future research.

II. From Language Models to Tool-Using Agents

Early LLMs excelled at generating human-like text but struggled with tasks that required external knowledge or precise operations. Two seminal works addressed this gap:

- **Toolformer:** This work showed that LLMs can teach themselves to use external tools via self-supervised learning, enabling them to decide which APIs to call and when to integrate their results into the text prediction process [1]
- **ToolLLM:** Building on the idea of tool use, ToolLLM enables LLMs to master over 16,000 real-world APIs, significantly expanding their operational scope and practical applicability [2]

These advancements marked a critical turning point, demonstrating that LLMs could transcend purely linguistic tasks to perform useful, real-world operations.

III. Large Action Models (LAMs)

While tool-using LLMs represent an important breakthrough, many applications require systems that can perform actions in dynamic environments rather than merely generating text. This led to the development of Large Action Models (LAMs), which are fine-tuned to execute tasks [3].

Developing and Training Large Action Models

Creating effective LAMs requires a systematic development pipeline that transforms a general-purpose language model into a specialized action execution system. Drawing on the insights from Wang et al. [3], this process encompasses several interconnected stages:

Data Collection and Preparation: The foundation of LAM development lies in gathering and curating task-specific data. This initial phase follows a two-phase approach:

- **Task-Plan Data Collection:** This involves collecting user requests and corresponding step-by-step plans. Sources include application documentation, WikiHow articles, and historical search queries. Each entry typically contains a task description and a detailed plan outlining the steps required to accomplish it.
- **Task-Action Data Collection:** This phase converts task-plan data into executable task-action data. Each plan step is transformed into concrete, actionable instructions that can be directly executed in the target environment. This process includes instantiation (adding specific operational details), execution validation, and evaluation to ensure correctness.

For example, a typical task-plan data entry might look like:

```
{
  "task_id": "powerpoint_task_001",
  "task": "Create a slide based on draft.docx"
    ,
  "plan": [
```

```
    "1. Open the draft.docx and read the
    content.",
    "2. Create a new PowerPoint file.",
    "3. For page 1, add ..."
  ]
}
```

While a corresponding task-action entry would include additional execution details:

```
{
  "task_id": "powerpoint_task_001",
  "task": "Create a slide based on draft.docx"
    ,
  "plan": [...],
  "actions": [
    {
      "step": "open the document",
      "controlLabel": "",
      "controlText": "",
      "function": "open",
      "args": {"file_path": "draft.docx"}
    }
  ]
}
```

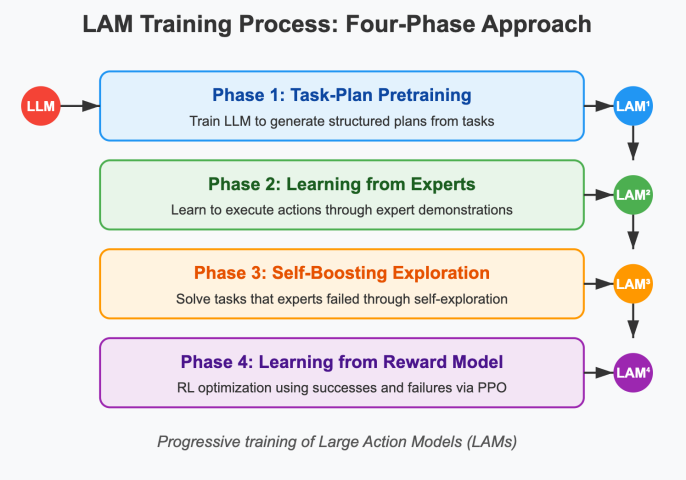

Fig. 1. Progressive four-phase training approach for transforming a general LLM into a specialized LAM.

Training an LLM into a LAM: A Four-Phase Approach: The transformation of a general-purpose language model into a specialized Large Action Model represents the technical core of LAM development. Wang et al. [3] propose a systematic, four-phase training process that progressively builds capabilities from planning to action execution:

1) **Task-Plan Pretraining:** In this foundational phase, the model (LAM^1) learns to generate structured, step-by-step plans for completing tasks. Using supervised fine-tuning with task-plan pairs, the model develops the ability to break tasks into logical sequences. At this stage, the model can generate coherent plans but lacks execution capabilities.
2) **Learning from Experts:** In the second phase, the model (LAM^2) learns to translate plans into concrete, executable actions through expert demonstrations. Training uses state-action pairs where each state includes the current UI environment and task context, and each action specifies which control to interact with and how. This phase grounds the model's reasoning in real application environments.
3) **Self-Boosting Exploration:** Moving beyond expert-only training, the third phase introduces self-improvement. LAM^2 attempts tasks that experts (like GPT-4o) failed to solve, generating new successful trajectories. These self-discovered solutions, combined with original expert demonstrations, create an augmented dataset used to train LAM^3. This represents a critical step toward greater model autonomy.
4) **Learning from a Reward Model:** The final phase introduces reinforcement learning, enabling the model to learn from both successes and failures. First, a reward model is trained on both successful and failed trajectories, assigning positive values to successful steps and negative values to failures. Then, LAM^4 is fine-tuned using Proximal Policy Optimization (PPO) on previously failed trajectories, guided by the reward model's feedback. This approach significantly enhances the model's decision-making abilities in complex scenarios.

Throughout this progressive training approach, each phase builds upon previous ones, creating increasingly capable systems that combine high-level planning, expert-guided execution, autonomous exploration, and reward-based optimization.

Integration and Grounding: Once trained, the LAM must be integrated into an agent framework to interact with real-world environments:

- **Environment Interface:** Mechanisms to observe and interact with the target environment, such as UI Automation APIs that provide information about actionable controls.
- **Action Execution:** Systems to translate model outputs into tangible actions within the environment, mapping predicted actions to specific function calls.
- **Memory Management:** Components to maintain historical actions and plans, providing essential context for future decisions and enabling more coherent multi-step execution.
- **Feedback Loops:** Structures to collect environmental responses and adapt accordingly, allowing the agent to respond to changing conditions.

Evaluation: The final stage involves rigorous assessment of the LAM's performance:

- **Offline Evaluation:** Testing on controlled datasets without environmental interaction, measuring metrics like Task Success Rate, Object Accuracy, and Operation Accuracy.
- **Online Evaluation:** Deploying the model in real environments to measure practical performance, including Task Completion Time and Average Step Latency—crucial for assessing real-world applicability.

This comprehensive development pipeline ensures that LAMs can effectively translate user intentions into meaningful actions within specific operational contexts. Each stage contributes essential capabilities, culminating in models that

can perform complex, multi-step tasks in real-world environments—representing a fundamental advancement over traditional language models limited to text generation.

Applications and Impact

LAMs enable a wide range of applications previously challenging for traditional LLMs, including:

- Autonomous interaction with software applications and operating systems
- Control of physical devices and robotic systems
- Complex task automation in specialized domains like healthcare or finance
- Multi-step problem-solving that requires environmental awareness

These capabilities mark a significant advancement toward artificial general intelligence (AGI), enabling AI systems to automate tasks that were previously only possible with human intervention.

LAMs thus represent a paradigm shift—from language generation to action execution—allowing AI systems to automate complex processes with minimal human intervention, substantially expanding their practical utility across numerous domains.

IV. Multi-Agent Systems

The next step in this evolution is the integration of LAMs into multi-agent systems (MASs). In such systems, multiple agents—each potentially powered by LAMs—collaborate to accomplish tasks that exceed the capability of any single agent [4].

- **Multi-Agent Collaboration Mechanisms:** A comprehensive survey of LLM-based MASs outlines the collaborative mechanisms, communication structures, and coordination protocols that allow multiple agents to work together efficiently.
- By endowing individual agents with the ability to perform actions (through LAMs), MASs can distribute complex tasks, dynamically allocate subtasks, and achieve higher fault tolerance and robustness.

This integration marks a significant leap in AI system design, enabling real-world applications where agents interact, negotiate, and cooperate autonomously.

V. Applications and Use Cases

Recent studies have demonstrated practical applications of these advances:

- **xLAM – A Family of Large Action Models:** The xLAM series exemplifies how LAMs can empower AI agent systems [5]. These models, spanning various sizes and architectures, have been shown to excel in executing real-world tasks, including tool use and interactive operations.
- **LLM-based Multi-Agent Systems: Techniques and Business Perspectives:** This work bridges the gap between academic research and practical applications [6], discussing dynamic task decomposition, proprietary data preservation, and monetization strategies.

Together, these efforts illustrate the maturation of AI systems from isolated language models to integrated, action-capable multi-agent frameworks.

VI. Challenges and Limitations

Despite these impressive advances, several challenges remain:

- **Coordination and Planning:** Ensuring effective task allocation and dynamic planning across multiple agents remains an open problem, as discussed in studies on multi-agent systems' challenges.
- **Memory and Context Management:** Maintaining coherent shared context and memory in MASs is critical to prevent cascading errors and ensure reliable performance.
- **Scalability and Robustness:** As systems grow in complexity, efficient scaling and robust fault tolerance become increasingly important.
- **Ethical and Safety Considerations:** The deployment of autonomous agents in real-world environments requires careful attention to safety, accountability, and ethical implications.

VII. Future Directions

Future research may focus on developing standardized frameworks for inter-agent communication, enhancing real-time adaptability, and integrating advanced security measures. Addressing these challenges will be key to realizing the full potential of AI systems capable of performing multiple productive tasks.

VIII. Conclusion

The evolution from LLMs to tool-using models, and ultimately to Large Action Models integrated within multi-agent systems, represents a fundamental shift in AI capabilities. By transitioning from passive text generation to active task execution, these systems are poised to transform numerous domains—from automated workflows to complex collaborative applications. While significant progress has been made, further research into coordination, scalability, and safety is essential to fully harness the power of these intelligent systems.

VIII. References

[1] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: Language models can teach themselves to use tools," *arXiv preprint arXiv:2302.04761*, 2023.

[2] Y. Qin, S. Liang, H. Ye, M. Zhu, C. Zhao, S. Li, Y. Chen, Z. Ding, C. Xu *et al.*, "Toolllm: Facilitating large language models to master 16000+ real-world apis," *arXiv preprint arXiv:2307.16789*, 2023.

[3] Z. Wang, Y. Gu, Z. Ding, Y. Jiang, X. Jiang, Y. Ding, Z. Zhao, Q. Zhao, Y. Zhao, Y. Dong *et al.*, "Large action models: From inception to implementation," *arXiv preprint arXiv:2402.01686*, 2023.
[4] L. Wang, W. Chen, J. Peng, S. Wu, Z. Xi, H. He, Y. Xie, J. Bian, and Y. Min, "A survey on large language model based autonomous agents," *arXiv preprint arXiv:2308.11432*, 2023.
[5] Y. Zhao, Z. Ding, Y. Gu, Z. Wang, Y. Jiang, X. Jiang, Z. Zhao, Q. Zhao, Y. Zhao, Y. Dong *et al.*, "xlam: A family of large action models," *arXiv preprint arXiv:2402.01086*, 2024.
[6] Y. Zhao, Y. Jiang, Y. Gu, Z. Wang, Z. Ding, X. Jiang, Z. Zhao, Q. Zhao, Y. Zhao, Y. Dong *et al.*, "Llm-based multi-agent systems: Techniques and business perspectives," *arXiv preprint arXiv:2311.05657*, 2023.