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].

From LLMs to LAMs: A Paradigm Shift

LAMs extend beyond the capabilities of traditional LLMs by bridging the gap between language understanding and action execution. As Wang et al. explain in "Large Action Models: From Inception to Implementation," LAMs are designed to interpret user intentions from diverse inputs, generate actionable plans, and interact directly with both digital and physical environments.

The transition from LLMs to LAMs represents a fundamental shift in AI systems—from passive text generation to active engagement with the world. While LLMs excel at generating human-like text based on user inputs, they lack the ability to directly interface with environments to execute actions. LAMs address this limitation by converting user requests into concrete, executable actions grounded in specific operational contexts.

Key Characteristics of LAMs

According to Wang et al., LAMs are distinguished by several advanced capabilities:

- **Interpretation of User Intentions:** LAMs can accurately interpret user intentions from diverse input forms, including natural language requests, voice commands, and visual inputs.
- **Action Generation:** Unlike LLMs that produce primarily textual outputs, LAMs translate user intentions into actionable steps executable within specific contexts, such

as GUI operations, API calls, physical manipulations, or code generation.
- **Dynamic Planning and Adaptation:** LAMs can decompose complex tasks into subtasks and further into specific action steps, adapting to environmental changes and replanning as necessary.
- **Specialization and Efficiency:** By focusing on particular domains, LAMs achieve greater accuracy and adaptability while potentially operating with smaller model sizes compared to general-purpose LLMs.

Development Pipeline for LAMs

Creating effective LAMs involves a systematic development process outlined by Wang et al. [3]. To understand the process of transforming a LLM into a LAM it is neccesary that we spend some time understanding the training process:

Data Collection and Preparation: Gathering and curating task-specific data, including user queries, environmental contexts, and corresponding actions. This initial phase involves a two-phase approach:
- *Task-Plan Data Collection:* Collecting data consisting of user requests and corresponding step-by-step plans, sourced from documentation, websites like WikiHow, and search queries.
- *Task-Action Data Collection:* Converting task-plan data into executable task-action data, where each plan step is transformed into concrete actions that can be directly executed in the target environment.

For example, a typical task-plan pair in the data collection phase might look like the following JSON structure:

```
{
  "task_id": "powerpoint_task_001",
  "task": "Create a slide base on draft.docx",
  "plan": [
    "1. Open the draft.docx and read the
    content.",
    "2. Create a new PowerPoint file.",
    "3. For page 1, add ..."
  ],
  "actions": [
    {
      "step": "open the document",
      "controlLabel": "",
      "controlText": "",
      "function": "open",
      "args": {"file_path": "draft.docx"}
    }
  ]
}
```

Model Training: This critical phase transforms a general language model into a specialized action model through a progressive four-phase training strategy:
- *Phase 1: Task-Plan Pretraining*
In this foundational phase, the model learns to generate structured, step-by-step plans for completing tasks. Using supervised fine-tuning with cross-entropy loss:

$$L_{SFT}(LAM_\theta^1) = \frac{1}{N}\sum_{i=1}^{N} L_{CE}(P_i^{pred}, P_i^{true}) \quad (1)$$

This establishes the model's ability to break tasks into logical sequences, creating LAM^1 with planning capabilities.
- *Phase 2: Learning from Experts*
Here, the model learns to translate plans into concrete, executable actions through expert demonstrations:

$$L_{SFT}(LAM_\theta^2) = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T_i} L_{CE}(LAM_\theta^2(s_t), a_t) \quad (2)$$

This phase grounds the model's reasoning in real application environments, enabling action execution.
- *Phase 3: Self-Boosting Exploration*
Moving beyond expert-only training, this phase introduces self-improvement by letting the model attempt tasks that experts failed to solve:

$$L_{SFT}(LAM_\theta^3) = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T_i} L_{CE}(LAM_\theta^3(s_t), a_t) \quad (3)$$

Though the objective function appears similar to Phase 2, the innovation lies in using an augmented dataset that includes novel successful solutions discovered by the model itself.
- *Phase 4: Learning from a Reward Model*
The final phase introduces reinforcement learning to enable the model to learn from both successes and failures. This involves two steps:
a) Building a reward model: $r_t = RM(s_t, a_t; \phi)$
b) Applying Proximal Policy Optimization (PPO):

$$L_{PPO}(LAM_\theta^4) = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T_i} \min\Big(\Big[\text{ratio}\times\hat{A}_t\Big], \Big[\text{clip}(\text{ratio}, 1-\varepsilon, 1 \quad (4)$$

Where ratio $= \frac{LAM_\theta^4(a_t|s_t)}{LAM_{\theta_{old}}^4(a_t|s_t)}$
This phase significantly enhances the model's decision-making abilities by incorporating feedback signals from both successful and failed attempts.

Integration and Grounding: Embedding the LAM within an agent framework equipped with tools, memory, and environmental interfaces. This involves:
- *Environment Interface:* Mechanisms to observe and interact with the target environment (e.g., UI Automation APIs).
- *Action Execution:* Systems to translate model outputs into tangible actions within the environment.
- *Memory Management:* Components to maintain historical actions and plans, providing context for future decisions.
- *Feedback Loops:* Structures to collect environmental responses and adapt accordingly.

Rigorous Evaluation: Testing the LAM's performance through:
- *Offline Evaluation:* Testing the model on controlled datasets to measure metrics like Task Success Rate, Object Accuracy, and Operation Accuracy without environmental interaction.

- *Online Evaluation:* Deploying the model in real environments to measure practical performance metrics including Task Completion Time and Average Step Latency.

This comprehensive development pipeline transforms general language models into specialized action models capable of executing complex tasks in real-world environments. Each phase builds upon the previous one, creating progressively more capable systems that combine planning abilities, expert knowledge, self-guided exploration, and reinforcement learning to achieve robust, adaptive performance.

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.