ChatGPT

Symbolic World Models in a Sandboxed Academic Knowledge Domain

Introduction

Building a **symbolic world model** for an academic knowledge domain means equipping an AI agent with an internal representation of academic concepts and their relationships – a kind of "mental map" of scholarly knowledge. Unlike a physical or perceptual world model (as used in robotics or game agents), this world is a *semantic* environment: a structured space of concepts, theories, and literature. For example, Concepedia's multilingual concept graph illustrates such an environment – an online **conceptual encyclopedia** that indexes academic concepts across languages, linking them to publications, authors, and related ideas [1]. An AI operating in this realm must understand symbolic entities (like technical terms, theories, or methods), navigate knowledge structures (ontologies, knowledge graphs, citation networks), and perform complex reasoning (e.g. mapping a research field or discovering connections between concepts). This report analyzes how to construct an AI system capable of **comprehending and reasoning over symbolic academic representations**, **using structured tools** (knowledge bases, glossaries, citation indexes), **planning complex research tasks**, and **learning or refining its knowledge recursively**. We draw on recent advances – including *structured world models* in AI research agents like **Kosmos (2025)** and *Recursive Language Models (RLM, 2025)* – as well as inspiration from AutoGPT-style planning agents and academic assistant systems. We also compare these approaches to "world models" in other domains to illuminate common principles. The goal is a comprehensive architectural and theoretical blueprint for a **symbolic academic world model**.

Symbolic vs. Neuro-Symbolic Reasoning in Academic AI

A key foundation is the distinction between purely symbolic systems and modern neuro-symbolic approaches. **Symbolic AI** traditionally uses explicit representations – e.g. ontologies of concepts, logical rules, knowledge graphs – and performs reasoning via logic or graph traversal. For instance, an academic concept graph with nodes like *"structural graph theory"* linked to related subfields and key papers provides a **discrete, human-interpretable model** of knowledge [2]. This can enable precise reasoning (e.g. following links in a citation network or taxonomy). However, purely symbolic systems struggle with ambiguity and learning from raw data. By contrast, **neural models** (like large language models, LLMs) excel at pattern recognition and language generation but **lack explicit knowledge grounding**, often leading to hallucinations or difficulty with long-tail scholarly facts.

Neuro-symbolic architectures aim to combine these strengths. They integrate neural networks with structured knowledge bases to improve reasoning and factual accuracy [3]. For example, an LLM can be augmented with a knowledge graph of scientific facts: the graph provides concrete entities/relations and can be queried for exact facts, while the LLM provides flexible understanding and natural language interface. Recent studies show that grounding language models in knowledge graphs can dramatically boost accuracy and reduce errors. One analysis noted that knowledge-augmented systems achieved **86.3% accuracy versus 32–75% for traditional LLMs** on certain tasks, and enterprises report large gains in

productivity from such integrations [4] [5]. In the academic domain, this means an agent that **consults a knowledge graph of concepts or a citation database** will be more reliable than one relying on parametric memory alone. The system we envision is inherently neuro-symbolic: it maintains a **symbolic world model** (graph-like memory of academic knowledge) while using powerful neural components (LLMs) for language understanding, pattern recognition, and decision-making.

The "World Model" Concept for Knowledge Environments

In reinforcement learning and robotics, a *world model* usually refers to an agent's internal model of its environment, used to predict outcomes or plan actions. For a self-driving car, this might be a map of the road and physics of motion; for a game-playing AI, it's a simulator of game states. **By analogy, a symbolic world model in an academic context is an internal knowledge base that the AI uses as its "environment" for reasoning and planning**. Instead of physical objects and locations, the world is composed of **academic entities**: concepts, topics, papers, authors, datasets, and their relationships (e.g. *"Concept A is a subfield of B"*, *"Paper X cites Paper Y"*, *"Researcher Z co-authored Paper Y"*, etc.). The agent's "actions" in this world include retrieving information, traversing links, adding new facts, or posing new questions.

Crucially, this world model serves as a **long-term memory** and context for the agent. Just as an embodied agent updates its world model with new observations, an academic agent updates its knowledge model with new insights from each tool use or reasoning step. For example, if the agent reads a paper or executes an analysis, it can record the key findings into its world model for future reference. This persistent memory is essential to handle the scale of academic knowledge. Whereas a standard LLM has a fixed context window (few thousand tokens) and tends to "forget" earlier content, a structured world model can accumulate information over **very long sessions**. *Kosmos (2025)*, a recent AI Scientist system, demonstrates this: **its core innovation is a structured world model acting as a long-term memory, allowing it to maintain coherence over tens of millions of tokens across hundreds of steps** [6]. Prior agents without such memory would quickly run out of context and lose track of the research objective [7]. By writing important results to an external symbolic store, the agent's knowledge *persists* beyond the transient context of the neural model.

In practice, implementing this world model could involve a combination of a **knowledge graph database** and other datastores. The world model might include:
- **Entities and Relations:** e.g. scholarly concepts, key terms, researchers, with labeled relationships (ontology of fields, citation links, influences). This resembles Concepedia's concept graph or a domain ontology.
- **Facts and Findings:** e.g. experimental results, hypotheses, or conclusions the agent has derived. For instance, if tasked with mapping a field, the agent might add a fact like "Technique A is more accurate than Technique B on Problem C (citing Paper 123)."
- **Open Questions or Goals:** outstanding issues the agent needs to explore. Maintaining a list of unsolved sub-problems or queries enables planning (the agent can pick one to tackle next).
- **Tool Outputs and Metadata:** results from code execution (e.g. a statistical analysis result), or a summary of a paper the agent read, along with provenance (source citation or code cell reference).

Notably, Kosmos uses exactly this kind of **structured memory**. In Kosmos, *"the world model is a database of entities, relationships, experimental results, and open questions, updated after every task"*, serving as a shared blackboard between its sub-agents [8]. Each cycle of operation, Kosmos' agents (a data-analysis agent and

a literature agent) contribute to the world model: after running a data experiment or reading new papers, they **write back structured outputs and citations into the world model** [9]. This persistent symbolic memory allowed Kosmos to read ~1,500 papers and execute 42,000 lines of code over a 12-hour run without losing track [10] – far beyond the capacity of a single-pass LLM. By the end of the run, the accumulated world model contains a rich network of evidence and insights, which Kosmos' synthesis module then traverses to compile a final report [11]. Every statement in the report is traceable to the world model entries (each linked to code or literature), ensuring transparency [11]. This design echoes the *blackboard architecture* in classical AI: multiple specialist processes (agents) post their findings to a common structured memory, from which higher-level conclusions are drawn.

Planning and Tool-Augmented Reasoning in a Symbolic World

With a world model in place, the agent needs a **planning mechanism** to decide how to achieve its goals using available tools and knowledge. Academic tasks like *literature mapping* or *taxonomy expansion* are inherently complex and multi-step. For instance, "map out the key subtopics and papers in quantum machine learning" might involve steps such as: finding major subtopics, for each subtopic finding seminal papers, identifying relationships or contrasts between approaches, and so on. No single query or model call can produce a detailed, accurate map without decomposition. Thus, the agent must **break down high-level tasks into sub-tasks**, decide an order of execution, and possibly revise the plan based on intermediate results.

Modern agent frameworks provide examples of such planning modules. **AutoGPT-style systems** (from early 2023 onward) pioneered prompting techniques where an LLM iteratively generates its next action (e.g. tool use or a reasoning step) and a self-critique, looping until the objective is met. In practice, however, naive AutoGPT often struggled with long or complex tasks – partly due to shallow planning and limited memory (it largely kept track of goals and tasks in the prompt, which becomes unwieldy) [7]. More recent approaches introduce explicit planning structures. One paradigm is the **"agentic graph"** approach, where an orchestrator module builds a directed acyclic graph of tasks and their dependencies [12] [13]. In such a system, each node is a subtask (assigned to a specialized agent or tool), and edges encode prerequisite relationships. The orchestrator can dispatch tasks in parallel when possible and merge results, dynamically adjusting the plan if new sub-tasks emerge [13]. This is well-suited to academic research: one could imagine nodes for "Search papers on X", "Run experiment Y on dataset Z", "Summarize findings about W", all connected in a dependency graph leading to the final goal. This graph-based planner enables **branching and merging of reasoning**, which is important if multiple hypotheses or avenues need exploration concurrently [14]. Kosmos itself uses a simpler form of planning: at each cycle it generates up to 10 candidate tasks based on the current world model and overall objective [9]. These are essentially the next steps the system might take (e.g. *"run a differential analysis on dataset X"* or *"search literature for connection between gene A and disease B"* [15]). The system then executes those tasks, updates the world model, and repeats. This iterative plan-refine loop gradually expands the knowledge until the stopping criteria (time or sufficient findings) are met.

Tool-augmented reasoning is central in executing these plans. Our academic world model agent must interface with various **external tools**: search engines or academic databases, a knowledge graph query system, a code execution environment (for data analysis or simulations), perhaps a mathematical solver or a translator for multilingual sources. The agent needs a form of **tool-use abstraction**, i.e. a way to decide *which* tool to use for a subtask and *how* to use it, based on the task requirements. One successful pattern is the ReAct framework (Reason+Act) where the LLM is prompted to produce either a thought (internal

reasoning) or an action (a tool call) at each step, interleaving them [16]. This can be extended with a toolkit and instructions for each tool's usage. For example, if the task is *"find all known methods that use Transformer models for few-shot learning"*, the agent might decide to: 1) query its **knowledge graph** for relationships (to get a preliminary list of methods and papers), 2) for each result, use a **retrieval tool** (RAG) to fetch paper summaries, then 3) use the LLM to compile the findings into a summary. This kind of hybrid strategy was demonstrated by Mutlu (2025), who built a research assistant that **extracts structured relations into a Neo4j knowledge graph, indexes papers into a vector store, and then answers queries by combining graph queries with retrieval-augmented generation** [17]. In their system, precise structured queries (e.g. "methods ↔ dataset ↔ metric" relationships) are handled by the graph for exact answers, while open-ended questions fall back on the RAG pipeline [18] [19]. The result is a more robust question-answering agent that leverages the **precision of symbolic queries** and the **coverage of neural retrieval**.

In our world model agent, we can generalize this idea. The agent's architecture can include a **Tool Selector module** (possibly implemented through prompt engineering or a simple policy model) that examines the current subtask and world model state to pick the best tool. For instance, if the subtask is recognized as a factual lookup (e.g. "What is the definition of concept X?"), the agent queries a glossary or knowledge graph. If it's an open research question ("Compare theory A and B"), it may retrieve relevant papers and then use the LLM to synthesize an answer. If it requires data processing ("Analyze dataset Y to test hypothesis Z"), it invokes a code execution environment. Each tool interaction both **reads from** and **adds to** the world model – e.g. the result of code analysis is added as a new data-fact node, or new references found are added as new literature nodes.

A sophisticated feature we aim for is the agent's ability to **replan and learn from intermediate results**. This means the planning is not a one-shot static plan; it's adaptive. As new information comes in, the agent might discover the problem is harder or different than anticipated, leading to new subgoals. Here the world model's queryable nature is key: the agent can continuously query its accumulated knowledge to decide next steps. This is akin to how a scientist refines experiments based on earlier findings. The *Agentic Learning Graph* concept formalizes this: it describes a process where an agent *iteratively expands and refines a knowledge graph*, feeding it back into its reasoning loop [20] [21]. In one example, starting from a seed concept, an LLM-driven agent added nodes and edges for related concepts or hypotheses in each iteration, and then used the growing graph as context to guide further exploration [22]. Over time, the graph developed meaningful structure (e.g. hub nodes that represent central topics) without being explicitly programmed – a form of **emergent learning of knowledge structure** [21]. This illustrates *recursive tool-assisted learning*: the agent uses the world model to decide what to learn next, learns it, updates the world model, and repeats.

Managing Long Contexts and Recursive Reasoning

Academic domains often involve **very long contexts** – reading dozens of papers or analyzing large corpora of text. A crucial challenge is ensuring the AI can handle such breadth without losing important details (the *context fragmentation* problem). Two complementary strategies have emerged to tackle this: (1) externalizing memory into the world model (which we discussed), and (2) employing *recursive reasoning mechanisms* to break down context processing. The **Recursive Language Model (RLM)** approach exemplifies the latter. An RLM is essentially a wrapper around a base LLM that allows it to **call itself (or another model) on sub-parts of the input, recursively, before producing a final answer** [23] [24]. By doing so, it creates the *illusion of an infinite context window*: the model processes chunks of input sequentially or hierarchically, rather than all at once [23] [25]. This was motivated by the observation of

"context rot", where an LLM's performance degrades as the prompt grows very long [26]. Instead of stuffing, say, an entire book or a giant list of papers into a single prompt, an RLM might read it chapter by chapter, summarizing or extracting relevant bits at each step and only passing necessary information forward.

Figure: A conceptual diagram of a Recursive Language Model (RLM) architecture. The RLM uses a root language model that can spawn sub-model calls within an external environment (like a Python REPL) to handle parts of a huge context, allowing it to process virtually unlimited text without losing coherence [23] [24].

In an academic research agent, RLM-style recursion can be invaluable. For example, if the agent's task requires reading 100 papers, a naive approach might hit context limits or cause the model to forget earlier papers by the time it reads the later ones. An RLM-enabled agent could instead treat the list of papers as an environment and query itself on subsets: "Summarize paper 1-10", "Summarize papers 11-20", then recursively synthesize: "Based on the summaries, what are the common themes?" and so forth. This is essentially how a human researcher might approach a large literature review – reading in manageable chunks and taking notes before writing an overall summary. Indeed, the RLM prototype by Zhang et al. found that a recursive strategy **outperformed even a larger single-pass model on challenging long-context tasks**, and could handle 10 million+ token contexts without performance degradation [27] [28]. This suggests that *scaling up reasoning by recursion* is a viable path for academic world models, complementing the world model memory.

Another way to implement recursive reasoning in planning is for the agent to **call itself as a sub-agent** on subproblems. This is related to the idea of *self-reflection* or *self-ask*. One agent variant, for instance, might decide: "To solve the big question, I need to answer sub-question A first – let me invoke an internal query to answer A." The agent then temporarily focuses on A (perhaps with a fresh context or specialized prompt), produces an answer, stores it, and then resumes the higher-level question. This hierarchical problem solving is natural in symbolic domains (e.g., proving a theorem by breaking it into lemmas). By equipping the agent with the ability to spawn such recursive subtasks (and possibly even learning when to do so), we achieve a form of *meta-reasoning*. The **Agentic Graph** orchestration mentioned earlier inherently supports this: a node in the task graph could itself be a complex problem that triggers another planning process within it [12] [13]. The combination of a structured world model and recursive reasoning allows the agent to tackle open-ended research problems in a **robust, scalable way** – it can always expand its knowledge further or dive deeper into a subtopic as needed, without being constrained by a single linear context.

Examples of Emerging Systems and Architectures

Kosmos (2025) – *an AI Scientist*: Kosmos is a state-of-the-art example of an academic discovery agent that implements many of the ideas above. Its multi-agent architecture (data analysis + literature search) and **persistent structured world model** enabled it to carry out 20-cycle research runs equivalent to *"6 months of a scientist's work"* [29] [30]. The world model in Kosmos shared information between the agents so that insights from data analysis (e.g. a new gene correlation found in an experiment) could immediately inform literature searches, and vice versa [31] [6]. Each piece of information was stored with provenance, allowing Kosmos to generate final reports where every claim is backed by a citation or code reference [32]. Notably, Kosmos' design addresses the coherence challenge that earlier systems like **FutureHouse's Robin** faced: *"Previous generations of AI Scientists… have been limited in the number of actions they can take before losing coherence"*, due to finite context windows [7]. Kosmos' use of a **structured world model** broke through that barrier, aggregating knowledge over thousands of intermediate steps without forgetting [10]. This shows the power of a well-designed symbolic memory. Furthermore, Kosmos illustrates planning in a

research context: it actively *formulates hypotheses and questions* each cycle based on the current knowledge state [9]. For example, after finding a clue in data, it might ask a new question like "is there a pathway linking this gene to that disease?" and then pursue it. Over cycles, this yields discoveries that even align with or reproduce human findings [33] [34] – a validation that the system's internal reasoning (its world model updates and task choices) are making sense scientifically.

Recursive Language Models (RLMs): While not a full agent on their own, RLMs offer a *technique* highly relevant to our architecture. The RLM concept can be integrated into the agent's LLM backbone to handle reading large knowledge sources. For instance, if our academic agent needs to incorporate a large textbook or an entire domain's literature into its world model, an RLM strategy would parse that content in chunks and recursively build an understanding. One can see RLM as providing a form of **dynamic reading comprehension module** that interfaces with the world model: it could populate the world model by reading source documents bit by bit, rather than requiring a human to pre-curate a knowledge graph. Early RLM experiments highlight that **decomposing context and querying an environment (like a scratchpad or REPL) leads to more accurate and efficient processing of long texts** [27] [25]. This aligns with the needs of a research agent that might read volumes of material and must avoid the "attenuation" of knowledge that happens in long prompt chains.

AutoGPT-style and Academic Agents: The original AutoGPT (2023) showcased the promise of let-Loose agent loops – an LLM generating its own tasks and executing them – but it also exposed pitfalls. Without structured memory or specialized knowledge, such agents often meander or repeat actions. New variants and related systems have tried to fix these issues. For example, some projects integrated **vector databases as long-term memory** for AutoGPT, so it could store and recall information outside the immediate prompt. Others introduced more deterministic planners or feedback mechanisms (e.g. the agent periodically reflects: *"Am I closer to the goal? If not, adjust strategy."*). In academic assistant applications, an "AutoGPT" would likely need heavy augmentation with domain knowledge. One emerging class of tools, sometimes dubbed *"research GPTs"*, combine LLMs with academic search engines. For instance, systems like **Elicit** (by Ought) use an LLM to formulate queries to Semantic Scholar and then summarize the retrieved papers, essentially acting as a literature review assistant. There are also closed-domain chatbots (e.g. trained on a specific corpus of scientific papers or an encyclopedic knowledge base) that can answer scholarly questions with citations. These are precursors to a full world-model agent; they typically handle one question at a time rather than a long research process. However, they underscore the importance of **tool integration** (search and retrieval) and factuality in academic AI. The **Causaly platform** is another real-world example: it built an AI research assistant for biomedicine that leverages a massive cause-effect knowledge graph of scientific findings [35]. By grounding queries in 500 million curated facts, it reportedly achieved a *90% reduction in manual literature review time* for users [35]. This is effectively a symbolic world model (the cause-effect graph) paired with an AI reasoning layer, specialized to a domain.

Moving forward, we see experimental agents that fully embrace *continuous learning* in their symbolic world. One could imagine an academic AI that not only uses the world model for one-off tasks, but **persists it across sessions** – effectively building up its own knowledge base over time, like a scholar accumulating expertise. The Agentic Learning Graph idea we cited is a step in that direction: the agent autonomously grows its knowledge network as it interacts with data [20] [21]. There are challenges here around verification (ensuring the agent doesn't insert false knowledge) and managing outdated or contradictory information. However, tools from symbolic AI like logical consistency checks, and from machine learning like embedding-based anomaly detection, could be employed to keep the world model's quality high.

Analogies to World Models in Other Domains

It's illuminating to compare the academic world model agent to agents in simulated or physical domains. In robotics and control, an agent with a world model might **plan a path** from point A to B, avoiding obstacles – it uses an internal map (the world model) to simulate movements. In our case, the "path" could be a chain of reasoning or a connection between two concepts. For example, planning a path in knowledge terms might mean finding a series of intermediate concepts that bridge a gap between two research areas (analogous to finding a route on a map) [36]. The obstacle is lack of information, which the agent circumvents by performing a focused information-gathering action (like a robot taking a detour). In both cases, a good world model helps the agent foresee consequences: a physical agent simulates "If I go this way, I'll slip on ice"; a knowledge agent simulates "If I pursue this sub-question, it might resolve the uncertainty in our hypothesis." This highlights how **planning and foresight** are common denominators of world models across domains.

Furthermore, techniques like Monte Carlo Tree Search (MCTS) used in game AI (AlphaGo, etc.) have a parallel in knowledge discovery: an agent could simulate different sequences of exploring topics (a kind of search tree through the space of questions) and evaluate which sequence yields the most insight. A form of this is *"Tree of Thoughts"* in LLM research, where the model explores multiple reasoning paths for a problem and chooses the best outcome. For an academic agent, one "thought" branch might correspond to following one hypothesis, and another branch exploring an alternative, with the world model keeping track of the state for each. This branching strategy can mitigate getting stuck in a dead-end (akin to a robot backtracking when hitting a wall).

The concept of **memory and state** is also analogous. An autonomous robot updates its state (position, environment map) as it moves. An academic agent updates its knowledge state as it reads or infers. The memory might be imperfect or partially observable – a robot might not see behind a wall; an agent might not have access to certain paywalled papers or might have knowledge gaps. In both scenarios, the agent must handle uncertainty. In robotics, Partially Observable Markov Decision Processes (POMDPs) address acting under uncertain state. In a knowledge model, we might need strategies for uncertainty like **active inquiry** – e.g. the agent explicitly recognizes a knowledge gap in its world model and formulates a new query to fill it (similar to a robot turning on a sensor or exploring a new area to gather missing information).

Finally, there is an analogy in how **learning** occurs. In simulation-based "world model" agents (e.g. Ha and Schmidhuber's World Models in 2018), the agent first learns a model of the environment's dynamics (often as a neural network) and then plans within that learned model. Our scenario is somewhat reversed: the world model is explicitly built (from curated knowledge and extracted facts), and the "dynamics" are the logical or semantic entailments between facts (for example, if $A \to B$ and $B \to C$, then $A \to C$ might be inferred in the knowledge graph). The agent can "learn" new dynamics by discovering new relationships (like learning a new causal link in a biomedical graph). This could be seen as analogous to a robot learning new physics – except here it's the physics of the knowledge domain (e.g. learning that a certain method tends to improve a certain metric might be a new rule it adds to its model). Thus, the world model approach in academic AI is a form of **continual learning** – as the agent engages in tasks, it refines its understanding of how concepts relate, which in turn makes it more effective in future tasks, much like how an embodied agent improves its environment model with experience.

Conclusion

Creating a symbolic world model for an academic domain involves weaving together **structured knowledge representation**, **tool-augmented reasoning**, **strategic planning**, and **iterative learning**. The systems emerging in 2025 – from Kosmos's multi-agent research loops to Recursive LM's context management and knowledge-graph-augmented agents – all point toward the feasibility of such an architecture. A successful academic world model will be one that, like a diligent scholar, **builds up a rich internal map of knowledge**, uses that map to navigate complex questions, and continuously updates itself with new findings. It will combine the **precision of symbolic reasoning** (e.g. exact queries and logical consistency) with the **flexibility of neural networks** (understanding nuance, generating language, spotting patterns). The theoretical foundations from both symbolic AI (ontologies, planning algorithms) and connectionist AI (deep learning, language models) are converging to support this. By drawing lessons from world models in robotics and other domains, we ensure our academic agent can plan and adapt in its own "conceptual universe." The end result envisioned is an academic AI that can, for example, take an open research problem and autonomously perform literature reviews, link theories, generate hypotheses, and even suggest experiments – essentially functioning as a sandboxed scientific **discoverer**. Such an agent could drastically accelerate understanding by exploring the conceptual world at a pace and breadth that no single human could, while still grounding its reasoning in the **symbolic structures of human knowledge** [30] [29]. The convergence of world modeling, planning modules, tool integration, and memory systems thus paves the way for the next generation of AI-powered academic research assistants – agents that truly *understand* and *navigate* the world of ideas.

Sources:

1. Zhang, A. L. (2025). *Recursive Language Models (RLMs)* – blog post [23] [24].
2. Sutter, M. (2025). *Meet Kosmos: An AI Scientist that Automates Data-Driven Discovery – MarkTechPost* [8] [9].
3. Edison Scientific (2025). *Announcing Kosmos – AI Scientist for Autonomous Discovery* – blog article [6] [7].
4. Berkowitz, J. (2025). *Kosmos: Accelerating Six Months of Science in a Single Day* – news blog [10] [30].
5. Sosa, J. F. (2025). *Graph RAG and Agentic Graphs* – technical whitepaper [36] [20].
6. Mutlu, A. (2025). *Research Assistant with Knowledge Graph + RAG* – Stackademic blog [17] [19].
7. Cappelloni, M. (2025). *Knowledge Graphs for AI Agents* – Medium article [35] [3].
8. **Concepedia** – *Multilingual Concept Graph Encyclopedia* [1] [2].

---

[1] (PDF) Intelligent Digital Libraries: From Retrieval to Recommendation
https://www.academia.edu/2751194/Intelligent_Digital_Libraries_From_Retrieval_to_Recommendation

[2] Structural Graph Theory - Concepedia
https://www.concepedia.online/concept/structural%20graph%20theory

[3] [4] [5] [35] Knowledge graphs for AI agents: The architectural backbone of next-generation intelligence | by Matteo Cappelloni | Medium
https://blog.hellord.com/knowledge-graphs-for-ai-agents-the-architectural-backbone-of-next-generation-intelligence-3878d02ca8f8?gi=ffed224df140

6 7 29 32 33 34 Kosmos: An AI Scientist for Autonomous Discovery
https://edisonscientific.com/articles/announcing-kosmos

8 9 11 15 Meet Kosmos: An AI Scientist that Automates Data-Driven Discovery - MarkTechPost
https://www.marktechpost.com/2025/11/09/meet-kosmos-an-ai-scientist-that-automates-data-driven-discovery/

10 30 Edison Scientific Kosmos: Accelerating Six Months of Science in a Single Day | Joshua Berkowitz
https://joshuaberkowitz.us/blog/news-1/edison-scientific-kosmos-accelerating-six-months-of-science-in-a-single-day-1709

12 13 14 16 20 21 22 36 Comparative Analysis of RAG, Graph RAG, Agentic Graphs, and Agentic Learning Graphs | by Jose F. Sosa | Medium
https://medium.com/@josefsosa/comparative-analysis-of-rag-graph-rag-agentic-graphs-and-agentic-learning-graphs-babb9d56c58e

17 18 19 Research Assistant Agent with Knowledge Graph + RAG | by Abdulvahap Mutlu | Stackademic
https://blog.stackademic.com/research-assistant-agent-with-knowledge-graph-rag-369ab84689b2?gi=d6e57c9b3542

23 24 25 26 27 28 Recursive Language Models | Alex L. Zhang
https://alexzhang13.github.io/blog/2025/rlm/

31 [2511.02824] Kosmos: An AI Scientist for Autonomous Discovery
https://arxiv.org/abs/2511.02824