JUXT AI Radar

An engineer's guide to the AI landscape from JUXT's CTO & AI chapter members

Q1 2026

Grid Dynamics

JUXT | A GRID DYNAMICS COMPANY

Introduction

2025 delivered on the promise of agentic AI. After years of bold predictions, we're now seeing genuine production value, and the impact is profound.

For software engineering, this represents the most significant transition since compilers changed what it meant to write code, but the implications reach even further. Agentic AI provides a co-intelligence that puts capabilities once requiring technical specialists within everyone's reach, much as spreadsheets did a generation ago: transformative business value and new categories of risk to manage. We're still early in understanding the full scope of this change, and this quarter's radar captures both the areas that are maturing and the emerging frontiers that point to what comes next.

On the maturity side, we've added Temporal to our Adopt ring, recognising that durable workflow orchestration has become essential infrastructure for agents that need to survive failures and run reliably over days rather than seconds. Organisations are moving beyond prototypes to ask harder questions: how do we understand our processes before automating them? How do we monitor agents in production? The new entries for process mining and LLM observability reflect this shift from experimentation to operation.

On the frontier side, we're seeing growing interest in approaches that extend beyond what LLMs alone can achieve. Ontologies provide grounded semantics, giving AI systems authoritative definitions and relationships rather than statistical associations. Neurosymbolic AI couples neural networks with symbolic reasoning, enabling explainable decisions and guaranteed rule compliance. World models build internal representations that simulate how environments behave, enabling systems to predict consequences rather than merely paraphrase scenarios from training data. Together, these approaches point toward AI architectures that combine LLM flexibility with the formal semantics that regulated industries require.

AI is also escaping the screen. Foundation models purpose-built for robotics and physical systems are maturing rapidly, and digital twin platforms such as NVIDIA Omniverse are enabling organisations to train and test AI in simulation before deployment. We've added coverage of both areas for teams working at the intersection of AI and the physical world.

As we enter 2026, this radar captures where the landscape stands today. We're still in the early stages of this transformation, one we're following with professional and personal interest.

— Henry Garner (CTO, JUXT), January 2026

Radar overview

Our radar visualises the AI landscape across two dimensions: quadrants divide the space by category, while radiants indicate our confidence in each technology.

Quadrants

The radar is divided into four quadrants, each representing a category of AI technology:

Languages & frameworks

The frameworks, libraries, and protocols that underpin AI development. These are the software foundations your applications are built with.

Techniques

Methodologies and practices for building AI systems: approaches such as RAG, prompt engineering, agent design patterns and evaluation methodologies. The "how" of AI development.

Tools

Software that enhances AI development workflows without being embedded in your application code: IDE extensions, CLI utilities, testing frameworks and observability solutions.

Platforms

Infrastructure and managed services that host and run AI workloads: cloud AI services, vector databases, model serving platforms and MLOps infrastructure.

Radiants

Each item is placed on one of four concentric rings representing our recommended adoption stance:

Adopt

Technologies we have high confidence in. Proven in production with strong community support; we'd recommend them for appropriate use cases without hesitation.

Trial

Worth pursuing. These show strong promise and are mature enough to invest time in learning and piloting. Consider them for new projects where you can absorb some risk.

Assess

Worth exploring to understand how they might affect your organisation. Research is warranted, but they're not yet ready for serious investment.

Hold

Proceed with caution. Technologies that are immature, overhyped, or being superseded. Not recommended for new work, though existing usage may be fine.

Contributors

The AI Radar is produced by JUXT's AI Chapter, drawing on our experience following industry developments and working with clients across sectors. The radar represents our current viewpoint, subject to change as the technology landscape evolves. We welcome feedback via LinkedIn, BlueSky, or email.

Henry Garner

Henry is JUXT's CTO and leader of the AI Chapter. He's implemented AI systems in domains as diverse as education, financial services, and local government in roles spanning data scientist, software engineer and CTO. He's author of the book Clojure for Data Science and maintainer of the open source statistics library kixi.stats.

Ben Halton

Ben is an account manager at JUXT with extensive experience engineering and architecting complex systems. His career spans domains from risk systems in Tier 1 banks to retail recommendation engines and wine trading platforms. His interest in AI is especially in how it can enhance developer experience and productivity.

Denis Lobanov

Denis is a software engineer at JUXT whose technical experience spans developing Linux kernel modules for Satellite communications to distributed graph databases to web backends. He is currently focused on providing a platform that integrates, controls and secures LLM interaction.

Oliver Marshall

Oliver is a software engineer at JUXT who specialises in building data processing systems and backend infrastructure. He's recently worked on integrating cutting-edge database technology for a client and approaches AI technologies with healthy skepticism: hopeful about what's possible but focused on what actually works in practice.

Neale Swinnerton

Neale is a principal engineer at JUXT. He's spent his career in software development across many domains. He sees himself as an engineer more than a scientist, advising teams how to use pragmatic workflows to improve developer productivity and joy.

Chris Williams

Chris is a software engineer at JUXT with broad experience across industries and technologies. He has long seen automated testing as a superpower for building reliable systems, and now views large language models as the next tool for boosting productivity while supporting real learning.

We're grateful to Rhi Hanger, whose insights informed our ontology coverage, Gio Kiladze, who authored the process mining section, and Lucio D'Alessandro, who built the tooling that generates this publication.

Radar at a glance

Languages & frameworks

ADOPT

1. PyTorch
2. dbt
3. MCP

TRIAL

4. AutoGen
5. A2A
6. DeepEval
7. LlamaIndex

ASSESS

8. Prolog
9. JAX
10. LangChain & LangGraph
11. OpenAI AgentKit
12. PydanticAI
13. Smolagents
14. CrewAI
15. DSPy
16. LinkML

HOLD

17. TensorFlow
18. Keras
19. R
20. OpenCL

Techniques

ADOPT

21. Classical ML
22. RAG
23. LLM-as-a-judge
24. BERT variants
25. Few-shot prompting
26. Agentic tool use

TRIAL

27. Cross-encoder reranking
28. Ontologies for AI grounding
29. Model distillation & synthetic data
30. UMAP
31. Claude Skills

ASSESS

32. Structured RAG
33. Neurosymbolic AI
34. World models
35. LLM reproducibility
36. Hypothetical document embeddings (HyDE)
37. Fine-tuning with LoRA
38. Physical AI and robotics foundation models

HOLD

39. Word2Vec & GloVe
40. t-SNE
41. Zero-shot prompting
42. Chain of thought (CoT)
43. AI pull request review

Radar at a glance

Tools

ADOPT

44. Software engineering copilots
45. Provider-agnostic LLM facades
46. Notebooks

TRIAL

47. MLflow
48. Vector databases
49. Local model execution environments
50. LLM observability tools

ASSESS

51. AI application bootstrappers
52. Visual computer use agents
53. Lakera
54. Structured output libraries

HOLD

55. Conversational data analysis

Platforms

ADOPT

56. Foundation models
57. Weights & Biases
58. Temporal
59. Data pipeline orchestration tools
60. Cloud model hosting platforms

TRIAL

61. Production AI monitoring platforms
62. Open weight LLMs
63. AI-powered workflow automation platforms
64. Digital twin platforms

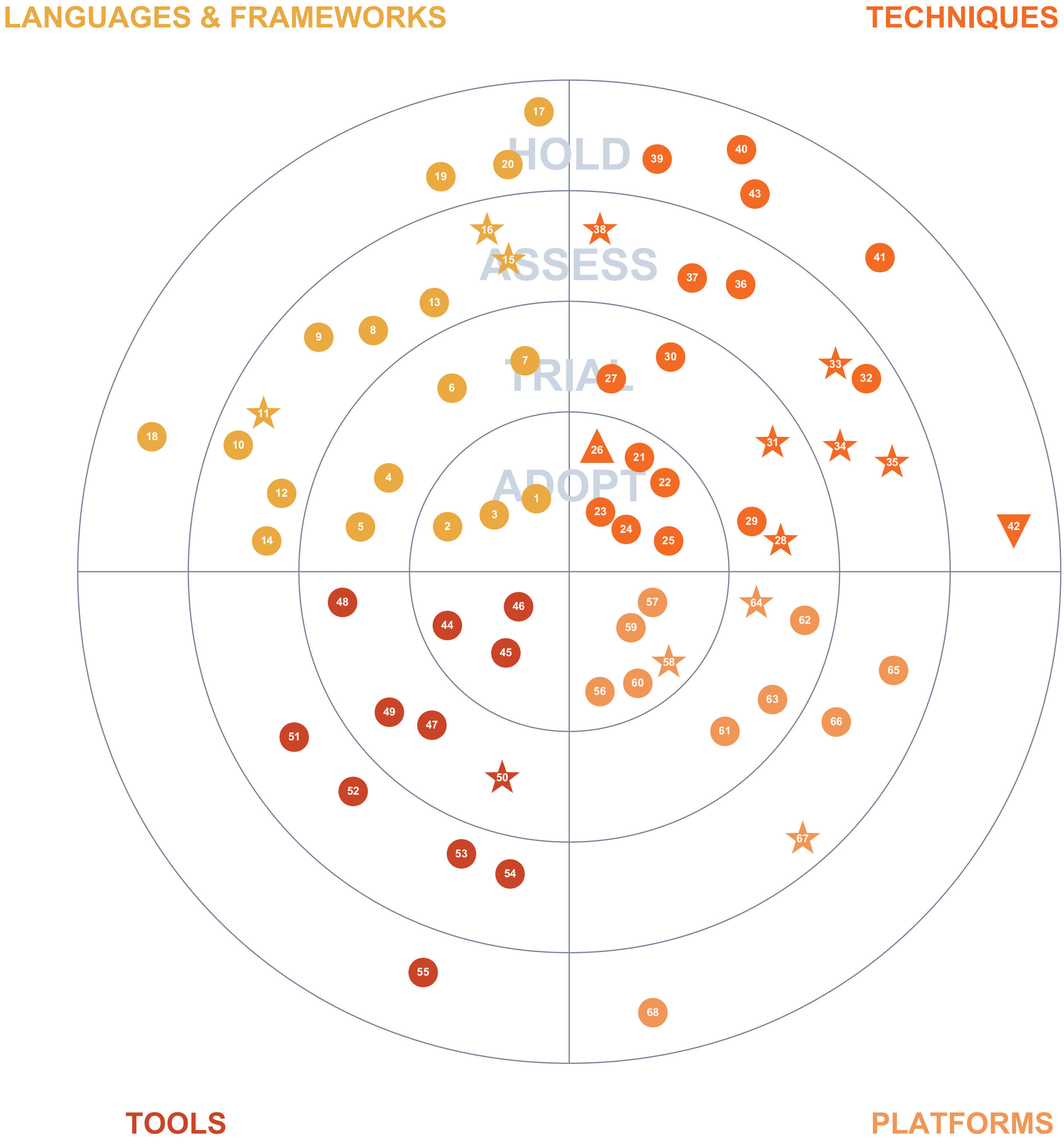
ASSESS

65. Galileo
66. Kubeflow
67. Process mining platforms

HOLD

68. Building against vendor-specific APIs

The Radar



● No change ▲ Moved up ▼ Moved down ★ New entry

Languages & frameworks

The frameworks, libraries, and protocols that underpin AI development. These are the software foundations your applications are built with.

Adopt

1. PyTorch
2. dbt
3. MCP

Trial

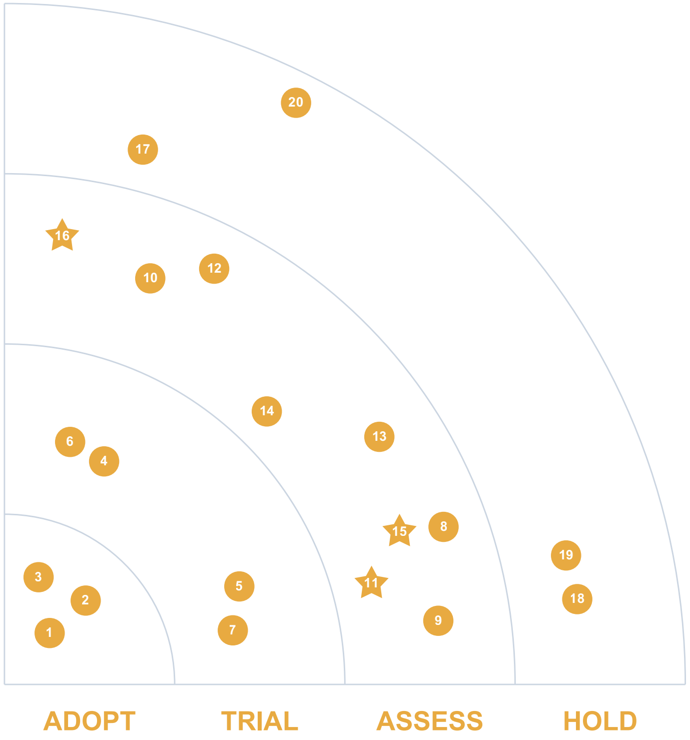
4. AutoGen
5. A2A
6. DeepEval
7. LlamaIndex

Assess

8. Prolog
9. JAX
10. LangChain & LangGraph
11. OpenAI AgentKit
12. PydanticAI
13. Smolagents
14. CrewAI
15. DSPy
16. LinkML

Hold

17. TensorFlow
18. Keras
19. R
20. OpenCL



Adopt

These languages and frameworks represent mature, well-supported technologies that are ready for production use. They offer excellent performance and proven track records in real-world applications.

PyTorch

PyTorch has demonstrated consistent maturity and widespread adoption across both research and production environments, earning its place in our Adopt ring. We're seeing it emerge as the default choice for many machine learning teams, particularly those working on deep learning projects, thanks to its intuitive Python-first approach and dynamic computational graphs that make debugging and prototyping significantly easier.

The framework's robust ecosystem, exceptional documentation and strong community support make it a reliable choice for teams at any scale. While TensorFlow remains relevant, particularly in production deployments, PyTorch's seamless integration with popular machine learning tools, extensive pre-trained model repository and growing deployment options through TorchServe have addressed previous concerns about production readiness. The framework's adoption by major technology organisations and research institutions, coupled with its regular release cycle and stability, gives us confidence in recommending it as a default choice for new machine learning projects.

dbt

We've placed dbt (data build tool) in the Adopt ring because it has proven to be an essential framework for organising and managing the data transformations that feed AI systems. dbt brings software engineering best practices such as version control and testing to data transformation workflows, which is crucial when preparing data for AI model training and inference.

The reliability and maintainability of AI systems heavily depend on the quality of their input data, and dbt helps teams achieve this by making data transformations more transparent and trustworthy. We've seen teams successfully use dbt to create clean, well-documented data pipelines that connect data warehouses to AI applications, while maintaining the agility to quickly adapt to changing requirements. Its integration with modern data platforms and strong community support make it a solid choice for organisations building out their AI infrastructure.

MCP

Anthropic's Model Context Protocol (MCP) has rapidly gained adoption since its introduction, addressing the critical need for standardised integration between language models and external tools. We've placed MCP in the Adopt ring based on its practical utility and straightforward implementation process.

MCP solves the persistent problem of connecting AI models to organisational data and tools without requiring custom integration work for each connection. The protocol's popularity stems from how straightforward MCP servers are to create and deploy, our teams have successfully built functional MCP servers within a matter of hours. This ease of implementation, combined with the growing ecosystem of community-created servers, significantly reduces development overhead.

For organisations evaluating MCP, the value proposition is clear: rather than building bespoke integrations between AI assistants and internal systems, teams can leverage existing MCP servers or create new ones following established patterns. The protocol handles context management and tool discovery effectively, enabling models to reason appropriately about available capabilities.

We recommend starting with existing MCP servers that match your requirements before building custom implementations. The protocol's design encourages reusability, meaning investments in MCP server development can benefit multiple AI applications across your organisation.

Since our last radar, we've continued to see rapid uptake of the MCP protocol within organisations. Some are pursuing ambitious goals of making all internal APIs AI-accessible via MCP servers, creating a unified interface through which AI assistants can interact with enterprise systems. This standardisation effort promises significant productivity gains, though teams should be realistic about the implementation investment required.

For simpler workflows that operate on local files and code rather than external services, Claude Skills offer a lighter-weight alternative worth considering before committing to MCP server development.

Trial

These languages and frameworks show promising potential with growing adoption and active development. While they may not yet have the same maturity as Adopt technologies, they offer innovative approaches and capabilities that make them worth exploring for forward-thinking teams.

AutoGen

We've placed AutoGen in the Trial ring based on its promising approach to orchestrating multiple AI agents for complex problem-solving. This Microsoft-developed framework enables developers to create systems where AI agents can collaborate, dividing tasks between specialised roles such as coding and reviewing, similar to how human development teams operate. While still evolving, we've seen compelling early results from teams using AutoGen to build more sophisticated AI applications, particularly in scenarios requiring multi-step reasoning or specialised domain knowledge.

The framework's ability to handle interaction patterns between agents with built-in error handling and recovery shows particular promise for enterprise applications. However, we recommend carefully evaluating its fit for your specific use case, as the overhead of managing multiple agents may not be justified for simpler applications where a single large language model would suffice. We're also watching how the framework's approach to agent coordination evolves as the field matures.

A2A

Google's Agent2Agent (A2A) protocol addresses the emerging need for standardised communication between AI agents in multi-agent systems. Launched in April 2025 and now governed by the Linux Foundation, A2A enables agents from different providers to discover each other's capabilities and collaborate on complex workflows without requiring custom integration work.

The protocol complements rather than competes with Model Context Protocol. Whilst MCP focuses on connecting AI models to tools and data sources, A2A specifically handles agent-to-agent communication. This distinction becomes important as organisations move towards multi-agent architectures where specialised agents collaborate to accomplish complex tasks requiring diverse capabilities.

A2A's design centres around "Agent Cards" that advertise capabilities in JSON format, enabling dynamic task delegation between agents. The protocol supports various modalities including text and video streaming, with built-in security features for enterprise deployment. Industry backing from over 150 organisations, including major hyperscalers, technology providers and consulting firms, suggests strong momentum for adoption.

We've placed A2A in Trial because whilst the protocol shows clear potential and has impressive industry support, it remains relatively new with limited production deployment patterns. Early implementations suggest promise for organisations building complex multi-agent systems, but teams should evaluate whether their use cases truly require agent-to-agent communication versus simpler architectures. For most organisations, starting with MCP for tool integration before exploring A2A for multi-agent scenarios represents a sensible progression path.

DeepEval

We've placed DeepEval in the Trial ring as it addresses a critical gap in AI application development: the systematic evaluation of Large Language Model outputs. While traditional software testing frameworks focus on deterministic outcomes, DeepEval provides a comprehensive toolkit for assessing the reliability and accuracy of AI-generated content.

The framework stands out for its practical approach to testing LLM applications, offering built-in metrics for evaluating responses across dimensions such as relevance and factual accuracy. What particularly impressed our committee was its ability to handle both unit and integration testing scenarios, making it valuable for teams building production-grade AI systems. However, we recommend starting with smaller, non-critical components first, as best practices around LLM testing are still emerging and the framework itself is relatively new to the ecosystem.

LlamaIndex

LlamaIndex, formerly known as GPT Index, is a framework that supports developers in connecting large language models with external data sources in a structured way. It provides tools to build indices, data structures that help LLMs access relevant information efficiently, thereby improving their ability to handle specific tasks requiring contextual or domain-specific data.

We consider LlamaIndex suitable for teams trialling methods to augment LLM performance, especially in data-centric applications. While its modular design and focus on customisation are appealing, its relative immaturity as a toolkit means that teams may encounter challenges around setup or adapting it to complex datasets. As with many emerging tools, its value depends on careful experimentation and matching it to the right problem space.

Assess

These languages and frameworks represent emerging or specialized technologies that may be worth considering for specific use cases. While they offer interesting capabilities, they require careful evaluation due to limited adoption or uncertain long-term viability.

Prolog

We've placed Prolog in the Assess ring due to its renewed relevance as a practical tool for neurosymbolic AI architectures. This decades-old logic programming language offers something LLMs fundamentally lack: guaranteed logical inference with explainable reasoning chains.

The value proposition is straightforward. LLMs excel at understanding natural language and recognising patterns, but they cannot reliably follow complex rules or explain why they reached a conclusion. Prolog does exactly this. By coupling an LLM with a Prolog reasoning engine, teams can build systems where the LLM handles ambiguous input and the Prolog component enforces business logic, validates conclusions, or traverses knowledge graphs. This combination has been likened to Kahneman's system 1 (fast, intuitive) and system 2 (slow, deliberate) modes of thinking.

While Prolog has been around since the 1970s, we are seeing renewed experimentation with it as a reasoning layer alongside modern LLMs. Implementations typically use Prolog to represent domain rules and relationships that the LLM queries or that validate LLM outputs before they reach users. This pattern is particularly valuable in regulated industries where decisions must be auditable and rule compliance is mandatory.

We've kept Prolog in Assess because the tooling ecosystem for LLM integration remains immature and performance can be challenging at scale. Teams should also consider whether semantic web technologies (RDF, OWL, SPARQL) might serve similar purposes with better tooling support. However, for organisations exploring neurosymbolic approaches, Prolog offers a well-understood foundation for symbolic reasoning that merits evaluation. At minimum, understanding what Prolog provides helps clarify what pure LLM approaches cannot deliver.

See also: Neurosymbolic AI, Ontologies for AI grounding

JAX

We've placed JAX in our Assess ring as we observe increasing interest in this ML framework that combines NumPy's familiar API with hardware acceleration and automatic differentiation. While TensorFlow and PyTorch remain dominant in the ML ecosystem, we're seeing JAX gain traction particularly in research settings and among teams working on custom ML architectures.

What interests us about JAX is its functional approach to ML computation and its ability to compile to multiple hardware targets through XLA (Accelerated Linear Algebra). The framework shows promise for projects requiring high-performance numerical computing, though we suggest careful evaluation of its relative immaturity in areas such as deployment tooling and the smaller ecosystem of pre-built components compared to more established frameworks. We recommend teams experimenting with JAX do so on research projects or contained proofs-of-concept before considering broader adoption.

LangChain & LangGraph

We've placed LangChain and its companion LangGraph in the Assess ring as they represent an emerging approach to building applications with Large Language Models. These frameworks provide structured ways to compose AI capabilities into more complex applications, with LangChain focusing on general-purpose AI interactions and LangGraph extending this to handle more sophisticated multi-step processes.

While these tools have gained significant adoption and show promise in reducing boilerplate code when working with LLMs, we recommend careful evaluation before widespread use. The rapid pace of change in the underlying AI platforms means that some of LangChain's abstractions may become outdated or less relevant as the ecosystem evolves. We've observed teams successfully using these frameworks for prototypes and smaller production systems, but also encountering challenges when requirements grow more complex or when they need to debug unexpected behaviours. Consider starting with focused experiments that test whether these tools truly simplify your specific use case rather than assuming they're the right choice for all AI development. Organisations invested in Microsoft's ecosystem should also consider Semantic Kernel, which offers similar orchestration capabilities with strong .NET support and Azure integration.

OpenAI AgentKit

OpenAI launched AgentKit at DevDay in October 2025, positioning it as "everything you need to build, deploy, and optimize agent workflows". The platform comprises Agent Builder for visual workflow design, ChatKit for embeddable agent interfaces, integrated evals for systematic testing and a Connector Registry for tool integration. Early partners including Albertsons, HubSpot and Bain & Company report significant efficiency gains, with Bain noting 25% improvements through the evaluation capabilities alone.

We've placed AgentKit in the Assess ring despite its high profile because several factors warrant caution. The platform is only months old, with key components such as Agent Builder still in beta. More significantly, AgentKit represents a substantial commitment to the OpenAI ecosystem. Unlike framework-agnostic alternatives such as LangChain or AutoGen, teams adopting AgentKit tie their agent infrastructure to a single provider's roadmap and pricing. Usage-based costs can become unpredictable as agentic workloads scale, particularly when agents make numerous tool calls or chain multiple LLM invocations.

For organisations already deeply invested in OpenAI's platform and comfortable with that dependency, AgentKit offers a streamlined path from prototype to production with strong tooling integration. However, teams requiring vendor flexibility or those uncertain about long-term OpenAI commitment should evaluate open alternatives first. The agent framework space remains highly competitive, and committing to a vendor-specific platform this early carries meaningful switching costs. We recommend treating AgentKit as one option among several rather than the default choice, assessing whether its conveniences outweigh the strategic implications of deeper vendor lock-in.

PydanticAI

We've placed PydanticAI in the Assess ring of our Languages & Frameworks quadrant because it represents a promising approach to building AI applications that merits closer examination, while not yet being broadly proven in production environments.

PydanticAI brings the well-regarded developer experience of FastAPI to generative AI application development. Built by the team behind Pydantic (which has become a foundation for many AI frameworks including OpenAI SDK, Anthropic SDK, LangChain, and others), it offers a familiar, Python-centric approach to building LLM-powered applications. The framework provides important features such as model-agnostic support across major LLM providers and structured responses through Pydantic validation, with a dependency injection system that facilitates testing.

What particularly interests us is how PydanticAI leverages existing Python patterns and best practices rather than introducing completely new paradigms. This could significantly lower the learning curve for developers working with AI. However, as a relatively new framework in a rapidly evolving space, we're placing it in Assess while we watch for broader adoption and production-proven implementations across different use cases. Organisations with Python-based stacks and teams familiar with FastAPI or Pydantic should consider evaluating PydanticAI for their AI application development needs.

Smolagents

We've placed smolagents in the Assess ring of the Languages & Frameworks quadrant based on our evaluation of its current state and potential.

This lightweight agent framework takes a minimalist approach with its core codebase of under 1,000 lines. Early feedback suggests it can be effective for quickly prototyping agentic concepts before transitioning to more robust frameworks such as AutoGen or LangGraph for production implementations. The framework's code-based agent approach, where agents execute actions as Python code snippets, reduces the number of steps and LLM calls in certain scenarios, though this comes with inherent security considerations.

We've positioned smolagents in Assess rather than Trial for several reasons: it lacks extensive production validation, the security implications of code execution require careful evaluation, and while benchmark results with models such as DeepSeek-R1 are interesting, we need to see more diverse real-world implementations. Teams exploring agent architectures should evaluate whether Smolagents' approach aligns with their specific needs and security requirements, whilst recognising its limitations for production-grade systems.

CrewAI

We've placed CrewAI in the Assess ring of the Languages & Frameworks quadrant because it represents a promising approach to multi-agent orchestration that's gaining traction among developers building complex AI systems.

CrewAI provides a framework for creating teams of specialised AI agents that work together to accomplish tasks through coordinated effort. Our team members report that it offers a well-structured approach to defining agent roles and task delegation: addressing many of the challenges involved in building effective agentic systems. The framework's emphasis on human-in-the-loop integration, along with the ability to combine specialised agents with different capabilities, makes it particularly valuable for complex workflows where single-agent solutions fall short.

While CrewAI shows significant promise and has already been used successfully in production environments, we've placed it in Assess rather than Trial because the multi-agent paradigm itself is still evolving. Organisations need to carefully evaluate whether the added complexity of managing multiple agents offers sufficient benefits over simpler approaches for their specific use cases. Teams should also be aware that best practices for agent collaboration are still emerging, and implementations may require considerable tuning and oversight to achieve reliable results.

DSPy

We've placed DSPy in the Assess ring as a promising approach to building LLM applications that treats prompts as optimisable programs rather than handcrafted text.

Developed at Stanford, DSPy shifts the paradigm from prompt engineering to programming. Instead of manually crafting prompts and hoping they work, developers define signatures (input-output specifications) and modules (composable building blocks). DSPy's optimisers then automatically generate effective prompts based on example data. This makes LLM pipelines more systematic and maintainable, addressing common complaints about the brittleness of handcrafted prompts.

The framework shows particular promise for complex pipelines involving multiple LLM calls, retrieval steps, or chain-of-thought reasoning. Teams report that DSPy's programmatic approach makes it easier to iterate on system behaviour without manually tweaking prompt text. The optimisation process can discover prompt strategies that humans might not have considered.

We've placed DSPy in Assess because the framework requires a different mental model than traditional prompt engineering and the learning curve can be steep. Teams should evaluate whether their use cases justify the investment in learning DSPy's abstractions. For complex multi-step pipelines where prompt optimisation would provide clear value, DSPy merits serious consideration. For simpler single-prompt applications, traditional approaches may remain more practical.

LinkML

We've placed LinkML in the Assess ring as a pragmatic approach to data modelling that bridges the gap between informal schemas and formal ontologies.

LinkML allows teams to define data models in YAML, a format accessible to developers without ontology expertise. From these definitions it generates multiple outputs: JSON Schema for validation, Python dataclasses for code, RDF/OWL for semantic web compatibility and documentation. This flexibility makes it valuable for phased ontology development, where teams want to start practically but preserve the option for formalisation later.

The framework emerged from biomedical informatics but applies broadly to any domain requiring structured data definitions. For AI applications specifically, LinkML models can define the entities and relationships that knowledge graphs should contain and the structured output schemas that LLMs should produce.

We've placed LinkML in Assess because adoption remains relatively niche and teams will need to evaluate whether the multi-output generation capability justifies learning a new modelling language. Organisations already committed to JSON Schema or existing ontology tooling may find less incremental value. However, for teams starting fresh on knowledge representation or ontology projects, LinkML offers an attractive middle path that avoids both the informality of ad-hoc schemas and the complexity of full OWL modelling.

Hold

These languages and frameworks are not recommended for new projects due to better alternatives or limited long-term viability. While some may still have niche applications, they generally represent technologies that have been superseded by more effective solutions.

TensorFlow

We have placed TensorFlow in the Hold ring for several reasons. While TensorFlow remains a capable deep learning framework that helped popularise machine learning at scale, we're seeing teams struggle with its steep learning curve and complex deployment story compared to more modern alternatives. The framework's syntax and intricate architecture could act as headwinds for teams new to machine learning.

PyTorch has emerged as the clear community favourite for both research and production deployments, with arguably a more intuitive programming model and better debugging capabilities. For new projects we recommend exploring higher-level tools or PyTorch unless there are compelling reasons to use TensorFlow, such as maintaining existing deployments or specific requirements around TensorFlow Extended (TFX) for ML pipelines.

Keras

We have placed Keras in the Hold ring primarily due to its transition from a standalone deep learning framework to becoming more tightly integrated with TensorFlow, along with the emergence of more modern alternatives that offer better developer experiences.

While Keras served as an excellent entry point for many developers into deep learning, providing an intuitive API that made neural networks more accessible, the landscape has evolved significantly. Frameworks such as PyTorch have gained substantial momentum, offering clearer debugging, better documentation and a more Pythonic approach. Additionally, recent high-level frameworks such as Lightning and FastAI provide similar ease-of-use benefits while maintaining closer alignment with current best practices in deep learning development. For new projects, we recommend exploring these alternatives rather than investing in Keras-specific expertise.

R

Despite R's historical significance in data science and statistical computing, we've placed it in the Hold ring for new projects. While R remains capable for statistical analysis and data visualisation, we're seeing its adoption declining in favour of Python's more comprehensive ecosystem for machine learning and AI workflows.

The key factors driving this recommendation are the overwhelming industry preference for Python-based ML frameworks and the stronger integration of Python with modern AI platforms and tools. While R retains some advantages for specific statistical applications and academic research, we believe teams starting new AI initiatives will benefit from standardising on Python to maximise their access to cutting-edge AI libraries and tools.

OpenCL

We've placed OpenCL in the Hold ring of our Languages & Frameworks quadrant. While OpenCL (Open Computing Language) was groundbreaking when introduced as a standard for parallel programming across different types of processors, we believe teams should look to alternatives for new projects.

Despite its promise of write-once-run-anywhere code for GPUs, CPUs, and other accelerators, OpenCL has seen declining industry support and faces significant challenges. Major hardware vendors have shifted their focus to more specialised frameworks such as CUDA for NVIDIA hardware, while newer alternatives such as SYCL and modern GPU compute frameworks offer better developer experiences with similar cross-platform benefits. The complexity of the OpenCL programming model, combined with inconsistent tooling support and a fragmented ecosystem, makes it increasingly difficult to justify for new development compared to more actively maintained alternatives.

Techniques

Methodologies and practices for building AI systems: approaches such as RAG, prompt engineering, agent design patterns and evaluation methodologies. The "how" of AI development.

Adopt

21. Classical ML
22. RAG
23. LLM-as-a-judge
24. BERT variants
25. Few-shot prompting
26. Agentic tool use

Trial

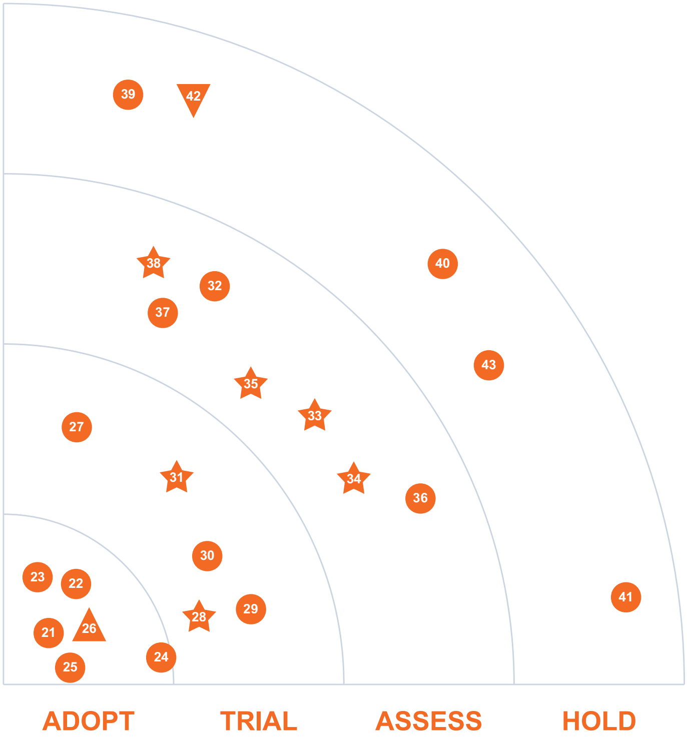
27. Cross-encoder reranking
28. Ontologies for AI grounding
29. Model distillation & synthetic data
30. UMAP
31. Claude Skills

Assess

32. Structured RAG
33. Neurosymbolic AI
34. World models
35. LLM reproducibility
36. Hypothetical document embeddings (HyDE)
37. Fine-tuning with LoRA
38. Physical AI and robotics foundation models

Hold

39. Word2Vec & GloVe
40. t-SNE
41. Zero-shot prompting
42. Chain of thought (CoT)
43. AI pull request review



Adopt

These techniques represent mature, well-supported approaches that are ready for production use. They offer excellent performance and proven track records in real-world applications.

Classical ML

We continue to see tremendous value in classical machine learning approaches such as random forests, gradient boosting (XGBoost, LightGBM), linear/logistic regression and support vector machines for many business problems. While attention has shifted dramatically towards deep learning and large language models in the last couple of years, these traditional techniques often provide the best balance of explainability and computational efficiency for structured data problems.

The key advantages that keep classical ML firmly in our Adopt ring include faster training times and lower computing requirements compared to deep learning approaches. However, it's important to recognise that realising these benefits requires both quality training data and staff with appropriate expertise. Unlike the recent wave of LLM-based solutions that have democratised AI capabilities for organisations without extensive data science teams, classical ML continues to demand specialised knowledge in feature engineering and model selection.

For organisations with the necessary data assets and technical capabilities, these methods work well even with the smaller datasets common in enterprise settings, often matching or exceeding the performance of more complex approaches while remaining more interpretable to stakeholders and easier to maintain. Their lower training costs and built-in feature importance metrics provide practical advantages that directly translate to business value, particularly as organisations face increasing pressure to make their ML systems both cost-effective and environmentally sustainable.

RAG

Retrieval-Augmented Generation (RAG) is an AI approach that combines search and text generation to produce more accurate responses. The approach helps prevent 'hallucination', cases where AI models confabulate plausible but incorrect information, by grounding responses in real data.

We're placing RAG in the Adopt ring because it addresses key challenges in deploying AI systems in information retrieval contexts. The technique is particularly valuable when accuracy and traceability of information are crucial, such as in customer service or compliance scenarios. While implementing RAG requires careful attention to document processing and embedding strategies, the widespread availability of tools and frameworks has significantly lowered the barriers to adoption. Teams should consider RAG as a foundational technique when building AI applications that need to leverage organisational knowledge.

We're particularly interested in monitoring how this technique develops alongside others improving AI system reliability and truthfulness. For example, by augmenting the approach with Self-RAG to recognise when more evidence needs to be gathered or responses refined for better accuracy. This 'self-criticism' mechanism has shown promising results in improving response quality and reducing confabulations.

See also Cross-encoder reranking, Structured RAG.

LLM-as-a-judge

We've placed LLM-as-a-judge in the Adopt ring because it has quickly proven itself to be one of the most practical and cost-effective techniques for evaluating AI system outputs. At first glance, it might seem like circular reasoning to have one LLM evaluate another LLM's work. However, the capabilities of today's strongest models are such that they can provide nuanced, multidimensional critique that simpler evaluation methods cannot match, except when using very constrained metrics such as exact match or BLEU scores (Bilingual Evaluation Understudy, a method for automatically evaluating machine translations).

This technique has become widely adopted in both offline and online evaluation scenarios. In offline evaluation, it scales far better than human assessment, allowing teams to test thousands of outputs quickly during development and quality assurance workflows. In online scenarios, an LLM judge can evaluate another LLM's output in real-time in production, enabling dynamic workflow adjustments or user experience modifications based on quality assessments. This real-time evaluation approach serves as a foundation for more sophisticated agentic workflows, where multiple AI components collaborate to refine outputs before user delivery.

Recent research demonstrates that the current frontier models can provide judgements that correlate strongly with human preferences across many common evaluation dimensions. For best results, we recommend using a different LLM as the judge than the one being evaluated, and viewing this approach as an augmentation to, not replacement for, human evaluation. The strongest LLMs can identify nuanced issues in reasoning and factuality that would otherwise require substantial human review time, creating a more efficient evaluation pipeline whilst preserving critical human oversight for final quality assurance.

BERT variants

Bidirectional Encoder Representations from Transformers (BERT) revolutionised Natural Language Processing (NLP) by allowing AI models to process human language by looking at words in relation to their entire context, rather than just left-to-right or right-to-left. Think of it like a reader who can understand a word by looking at all the surrounding words for context, rather than reading sequentially. The original BERT spawned a family tree of variants, with ModernBERT representing the latest evolution. Released in late 2024, ModernBERT improves legacy BERT through architectural updates which shorten training times and improve accuracy.

BERT-style models serve fundamentally different purposes than generative models such as GPT. While GPT models excel at generating text and conversational interactions, BERT models are optimised for understanding and analysis tasks such as classification and sentiment analysis. They're particularly valuable for creating semantic vector embeddings that capture text meaning in numerical form, making them essential components in Retrieval Augmented Generation (RAG) systems. In these pipelines, BERT embeddings help retrieve relevant information that is then fed as text to GPT models for generation: the models don't directly share embeddings, but rather work in complementary roles.

We particularly recommend DeBERTa for organisations starting new NLP projects. It handles word relationships more effectively using a disentangled attention mechanism and enhanced position encoding. DistilBERT is smaller and faster whilst retaining most of the model's performance, so it is particularly valuable for production deployments where latency requirements are strict or computing resources are limited, such as edge devices or high-throughput API services.

For organisations choosing between BERT and GPT models, consider your specific use case: BERT models require fewer computational resources for inference and excel at precise understanding tasks, while GPT models offer impressive out-of-the-box generation capabilities through accessible APIs. Many sophisticated AI applications today use both types in complementary roles, BERT for understanding and information retrieval, and GPT for generation based on that understanding.

There are options for specialised domains such as biomedical (BioBERT) or financial text (FinBERT). While these can outperform general models in their niches, they often require significant expertise to use effectively and may need additional tuning for specific use cases.

Few-shot prompting

The technique of providing examples to guide an AI model's responses has proven consistently effective across different Large Language Models. By showing the model a few examples of desired input-output pairs, developers can achieve more reliable and contextually appropriate responses without resorting to complex prompt engineering or fine-tuning.

The landscape is shifting. As models become more capable, interactive multi-turn approaches are gaining favour for complex tasks: rather than providing examples upfront, practitioners increasingly prompt models to ask clarifying questions and iterate toward a solution. This collaborative pattern often produces better results than static few-shot prompts, particularly in agentic workflows where the model can refine its approach based on feedback.

However, few-shot prompting retains an important role in non-interactive contexts. System prompts and automated pipelines don't afford the opportunity for clarifying dialogue. Here, well-chosen examples remain the most effective way to establish output format and domain conventions. We typically see diminishing returns beyond 3-5 examples, and the main trade-off remains token consumption.

Agentic tool use

We've moved agentic tool use to the Adopt ring for local, sandboxed environments. AI coding assistants that can edit files, run tests, execute shell commands and perform web searches deliver considerably more value than those limited to conversation, and the productivity gains we're seeing are substantial.

The ecosystem has matured to support this. Standards such as Model Context Protocol and OpenAI's Function Calling provide reliable integration patterns, while improved observability tooling means teams can monitor what their agents are actually doing. The Development Containers specification and tools such as devcontainer.ai make it straightforward to isolate agent execution, limiting the blast radius should anything go wrong.

The risks magnify significantly for applications that accept input from external users. Prompt injection attacks, where malicious content manipulates an agent into misusing its tools, remain an unsolved problem. An agent that can safely edit files for a developer becomes a serious liability when processing untrusted input from the internet. Bridging trusted and untrusted contexts requires careful security architecture: strict input validation, output verification, rate limiting and sandboxed execution with principle of least privilege.

Our recommendation is nuanced: adopt agentic tool use for local developer tooling and internal workflows where inputs are trusted, but proceed with caution for customer-facing systems, treating each tool permission as a potential attack vector.

Trial

These techniques show promising potential with growing adoption and active development. While they may not yet have the same maturity as Adopt techniques, they offer innovative approaches and capabilities that make them worth exploring for forward-thinking teams.

Cross-encoder reranking

Cross-encoder reranking sits in our Trial ring as a promising enhancement for AI search and chat systems. It works alongside traditional embedding-based search (where documents and queries are converted into numbers that represent their meaning) by taking a closer look at the initial search results. While embedding search is fast and good at finding broadly relevant content, cross-encoder reranking excels at understanding subtle relevance signals by looking at the query and potential results together.

Most teams we've observed use this as a two-step process: first, a quick embedding search finds perhaps 50-100 potentially relevant items from their knowledge base. Then, cross-encoder reranking carefully sorts these candidates to bring the most relevant ones to the top. While this additional step does add some processing time, we're seeing it deliver meaningful improvements in result quality across various use cases.

The technique has shown consistent improvements across different domains and use cases, often reducing confabulations in downstream LLM responses by ensuring higher quality context selection. Implementation has also become more straightforward with libraries such as sentence-transformers providing ready-to-use models. However, teams should be mindful of the additional latency introduced by the reranking step and may need to tune the number of candidates passed to the re-ranker based on their specific performance requirements.

Ontologies for AI grounding

As AI systems scale beyond isolated experiments, organisations are discovering that shared meaning becomes critical infrastructure. Ontologies provide what LLMs lack: authoritative definitions of entities and relationships that don't shift with statistical probability. They improve accuracy by grounding responses in agreed definitions. They enable knowledge graph traversal that pure RAG cannot achieve, following relationships between concepts rather than just retrieving similar text. And they support the structured outputs and tool definitions that agentic systems require.

Traditional ontology development tends toward two failure modes. Academic approaches aim for formal completeness using description logic and OWL, often spending years modelling before delivering value. Pragmatic approaches create spreadsheet taxonomies that grow organically but become unmaintainable as teams drift into inconsistent terminology. The key is to start lightweight and formalise selectively: begin with the core concepts that matter most and add formal semantics only where reasoning provides demonstrable value.

Not everyone agrees ontologies are the right grounding mechanism for the LLM era. Mark Burgess, creator of CFEngine and Promise Theory, argues that traditional ontologies impose rigid hierarchies that don't match how language models represent meaning, and proposes alternative graph structures designed to work with vector embeddings rather than impose categories on top of them. We're watching this debate with interest, but for organisations needing to ground AI in existing domain knowledge today, ontologies offer a practical path with mature tooling.

Graph databases such as Neo4j provide accessible implementation options, while LinkML offers a YAML-based modelling approach that maintains formalism without requiring deep ontology expertise. We recommend starting with a painful, high-value domain rather than attempting to model the entire organisation.

See also: LinkML, Neurosymbolic AI, Prolog

Model distillation & synthetic data

We've placed Model Distillation in the Trial ring of our Techniques quadrant. Distillation involves training a smaller, more efficient model to mimic a larger one. A common emerging pattern we're seeing is using LLMs to generate synthetic training data for this smaller model. The larger LLM acts as a "teacher," creating diverse, high-quality examples that can help the "student" model learn the desired behaviour. For instance, a large model might generate thousands of question-answer pairs that are then used to train a more compact model for a specific domain.

This creates an interesting synergy: the large LLM's ability to generate varied, nuanced responses helps create richer training datasets than might otherwise be available, while distillation makes the resulting solutions more practical to deploy. This approach makes AI deployment more practical and cost-effective, especially for edge devices or resource-constrained environments. However, we're keeping it in Trial as the process still requires considerable expertise to execute well. Teams need to carefully validate the quality of generated training data and ensure the distilled model maintains acceptable performance levels. There's also ongoing debate about potential amplification of biases or errors through this approach.

Be sure to check the licence of the model you're using for distillation. Llama forbids the use of its output to train other models. The launch of DeepSeek R1 in January 2025 brought distillation into popular consciousness, as it has been widely assumed that it represents a distillation of existing foundation models.

UMAP

UMAP (Uniform Manifold Approximation and Projection) enters our Trial ring as a promising dimensionality reduction technique that's gaining traction in the AI community. While t-SNE has been the go-to choice for visualising high-dimensional data, UMAP offers better preservation of global structure and runs significantly faster, making it particularly valuable for large-scale AI applications such as exploring embedding spaces and analysing neural network activations.

We're seeing successful applications across AI projects, especially for understanding LLM behaviours and exploring semantic relationships in vector spaces. Teams should invest time understanding UMAP's parameters, which require careful tuning to avoid misleading visualisations.

The Python UMAP library provides extensive documentation and explanation, with implementations also available for Rust, Java and R.

Claude Skills

We've placed Claude Skills in the Trial ring based on our positive experiences using them to bring structure and consistency to a variety of AI-assisted tasks. Skills are reusable prompt templates that codify workflows and domain expertise into repeatable patterns that AI coding assistants can follow.

Our teams have found Skills particularly valuable for drafting proposals, structured debugging workflows, generating commit messages and writing PR descriptions. The common thread is tasks that benefit from a consistent approach and structured output, but don't require access to external data or systems.

Skills provide a much simpler solution than MCP servers for many problems. Where MCP requires implementing a server and managing the protocol lifecycle, Skills are essentially markdown files that encode expertise directly. For teams wanting to standardise how AI assists with internal processes or ensure consistent output formats, Skills offer immediate value with minimal setup.

However, Skills don't replace MCP servers when integration with external services is required. If your workflow needs to query live databases or make authenticated API calls to internal services, MCP remains the appropriate choice. Skills work well with data that exists as files in your project or filesystem, since the AI assistant can already read those. MCP extends reach to running services and systems beyond what filesystem access provides.

We recommend teams start by identifying repetitive tasks where consistency matters, then experiment with Skills before investing in MCP server development.

Assess

These techniques represent emerging or specialized approaches that may be worth considering for specific use cases. While they offer interesting capabilities, they require careful evaluation due to limited adoption or uncertain long-term viability.

Structured RAG

Structured RAG extends basic RAG by organising knowledge in a more formal way, rather than just as chunks of text. Think of it like the difference between a filing cabinet (basic RAG) and a well-designed database (structured RAG). Instead of just retrieving text fragments, structured RAG can work with specific fields and relationships in your data. For example, in a product catalogue, it could separately track and retrieve product names, prices, specifications and reviews, understanding how these elements relate to each other.

The key advantages we're seeing in real-world applications include more consistent outputs and reduced confabulation rates compared to traditional RAG approaches. While implementations can vary, successful patterns are emerging around using JSON schemas or database-like organisations for retrieved information.

However, implementing structured RAG requires more upfront work in data organisation and schema design than traditional RAG. Teams need to carefully consider their data structures and retrieval patterns. This additional complexity is why we've placed it in Assess rather than Trial: while the benefits are clear, implementation patterns are still evolving.

Neurosymbolic AI

We've placed neurosymbolic AI in the Assess ring because it represents an architectural pattern that addresses fundamental limitations of pure large language model approaches. This is particularly relevant for regulated industries where explainability and rule compliance are non-negotiable.

The core idea combines neural networks with symbolic AI: neural networks excel at pattern recognition and handling ambiguity, while symbolic AI provides logical reasoning and explainable inference. LLMs understand natural language and recognise patterns well, but they cannot guarantee rule compliance or explain their reasoning in auditable ways.

The root issue is architectural. LLMs operate through probabilistic pattern matching over language, not causal modelling. They can reproduce explanations they have encountered but cannot reason about novel causal relationships. As Mark Burgess argues in his work on semantic spacetime, language models can only "paraphrase intentional knowledge": they predict what words typically follow a question about causes, rather than tracing actual causal chains. To get precise answers to precise questions requires systems that explicitly encode what causes what.

Neurosymbolic approaches couple LLMs with explicit knowledge structures and reasoning engines to address these gaps.

Financial services organisations find particular value in this pattern. Regulatory rules are non-negotiable constraints, not suggestions a model can approximate. Risk models need to know what entities are and how they relate, not just what words appear near each other. Compliance requires explainable decision trails that cannot be satisfied by asserting that the model produced a particular output. Similar pressures apply across regulated sectors including healthcare and insurance.

Practical implementations range from lightweight approaches through to sophisticated architectures. On the simpler end, teams constrain LLM outputs to valid ontology terms or use knowledge graphs to ground RAG retrieval. More advanced implementations use symbolic reasoning engines to validate or guide LLM-generated conclusions. Semantic Kernel provides orchestration capabilities in this direction. Renewed interest in Prolog reflects exploration of logic programming alongside LLMs.

We've placed this in Assess rather than Trial because production patterns are still emerging and the tooling remains immature. However, forward-thinking organisations in regulated sectors should be experimenting now. The combination of LLM flexibility with symbolic rigour is increasingly necessary as AI moves from assistive tools to autonomous decision-making, where approximate correctness is insufficient.

See also: Prolog, Ontologies for AI grounding, Agentic tool use, World models

World models

We've placed world models in the Assess ring because they represent an emerging alternative to pure language model architectures for tasks requiring causal reasoning and planning. Where LLMs predict the next token based on statistical patterns in text, world models build internal representations of how environments behave, enabling systems to simulate outcomes before acting.

The field is developing along several distinct paths. Yann LeCun's Joint Embedding Predictive Architecture (JEPA) learns by predicting missing information in an abstract embedding space rather than reconstructing raw pixels or tokens. Meta's V-JEPA and the recently released VL-JEPA extend this to video and vision-language tasks, achieving strong performance with significantly fewer parameters than autoregressive alternatives. LeCun's new venture, Advanced Machine Intelligence Labs, signals substantial investment in this direction.

Karl Friston's active inference framework, implemented by Verses AI in their AXIOM system, takes a different approach rooted in how biological systems model and interact with their environments. Rather than chasing reward signals, active inference agents build generative models of their world and act to minimise prediction error. Verses reports significant efficiency gains: a recent demonstration showed 60% performance improvement while using only 3% of the compute required by comparable deep learning approaches.

Generative world models form a third strand, with NVIDIA Cosmos and Google DeepMind's Genie 3 creating physically plausible simulated environments for training robots and autonomous systems. These overlap with the World Foundation Models discussed in our Physical AI entry.

For financial services, MarS from Microsoft Research demonstrates the pattern applied to market simulation. MarS uses a Large Market Model trained on order-level data to generate realistic, interactive market scenarios for forecasting, anomaly detection, market impact analysis and training trading agents without real capital at risk. The paper was accepted at ICLR 2025 and represents a concrete example of world models addressing problems that statistical approaches struggle with: simulating how markets respond to interventions rather than merely predicting price movements from historical patterns.

The relevance to enterprise AI lies in what these approaches offer that LLMs cannot: genuine causal modelling rather than statistical pattern matching over language. An LLM asked "what happens if I do X?" can only paraphrase similar scenarios from its training data. A world model can simulate the consequences. For applications in robotics and decision support where actions have real-world consequences, this distinction matters.

For teams wanting to experiment, several options are freely available. Meta's V-JEPA 2 and I-JEPA models are on HuggingFace under permissive licenses, with straightforward integration via the Transformers library. NVIDIA Cosmos models are openly available under the NVIDIA Open Model License, which permits commercial use. Verses AI publishes AXIOM code under an academic license, with their commercial Genius platform available for enterprise deployment. This remains an emerging area, and teams pursuing production implementations should be prepared for the additional engineering effort that comes with adopting less mature technology. That said, the barrier to experimentation is now low enough that forward-looking organisations can begin building practical familiarity with the paradigm.

See also: Neurosymbolic AI, Physical AI and robotics foundation models

LLM reproducibility

Large language models are non-deterministic even when configured for greedy sampling at temperature zero. This presents a fundamental challenge for regulated industries where Model Risk Management (MRM) frameworks require reproducible, auditable decision-making. The Financial Stability Board requires "consistent and traceable decision-making," while banking regulations such as OCC/SR 11-7 assume a level of model stability that generative AI does not provide.

The root cause extends beyond floating-point arithmetic. Recent research demonstrates that batch-dependent kernel operations cause outputs to vary with server load rather than input alone. Studies of cross-provider consistency found larger models showed as low as 12.5% consistency across identical requests, described as "fundamental architectural incompatibility with financial compliance requirements".

Organisations in regulated sectors have several options. For determinism-critical applications, smaller open weight models deployed on controlled infrastructure tend to achieve more reproducible outputs than larger models served via shared APIs. Where stochastic behaviour is acceptable, the variation must be well-characterised and bounded so it can be explained to regulators as a designed property rather than an infrastructure artefact. Regardless of approach, prompts and model versions should be treated as versioned code with change control and rollback procedures. Material changes should trigger revalidation against golden test sets.

For teams requiring determinism from larger models, SGLang now offers deterministic inference building on batch-invariant operators, reducing performance overhead to around 34% with CUDA graph optimisations. The underlying research was selected for oral presentation at NeurIPS 2025, signalling growing confidence that determinism is an engineering challenge rather than an inherent hardware limitation.

We've placed this in Assess because adoption of these techniques in production remains early. The tooling exists, but integrating deterministic inference into existing MLOps pipelines and explaining the approach to regulators require organisational investment. Teams subject to MRM requirements should be actively evaluating their options now rather than waiting for the ecosystem to mature further.

See also: Neurosymbolic AI, LLM-as-a-judge

Hypothetical document embeddings (HyDE)

We've found HyDE (Hypothetical Document Embeddings) to be an elegant solution to a common problem in search systems: their tendency to perform poorly when searching content that differs from their training data. HyDE works by first asking a large language model to imagine what an ideal document answering the user's query might look like. This 'hypothetical document' helps bridge the gap between how users naturally ask questions and how information is actually written in documents.

The system creates several of these imagined documents to capture different ways the answer might be expressed. These are converted into numerical representations (embeddings) and blended together. This averaged representation is then used to find real documents that are mathematically similar, which often leads to more relevant search results than traditional methods. The approach has proven particularly effective as part of larger systems, such as RAG (Retrieval Augmented Generation), where accurate document retrieval is crucial for generating reliable responses. Teams should evaluate HyDE particularly for cases where high-precision retrieval is crucial and the additional latency is acceptable.

See also: RAG, BERT

Fine-tuning with LoRA

We have placed Low-Rank Adaptation (LoRA) in the Assess ring. LoRA represents a significant advancement in making AI model customisation more practical and cost-effective. Rather than adjusting all parameters in a large language model (which can number in the billions), LoRA adds a small set of trainable parameters while keeping the original model unchanged. Think of it like teaching an expert to adapt to your specific needs without having to retrain their entire knowledge base. This approach typically reduces the computing resources needed for customisation by 3-4 orders of magnitude while maintaining most of the performance benefits of full fine-tuning.

The technique has proven its value across numerous enterprise applications, and robust tools such as Lightning AI's lit-gpt and axolotl have emerged to support implementation. However, we place it in the Assess ring rather than Trial because successfully applying LoRA still requires significant machine learning expertise and careful consideration of training data quality. Additionally, we recommend organisations view fine-tuning (including with LoRA) as a long-term strategy rather than a short-term investment. Fine-tuning typically ties you to a specific model architecture, and given the rapid pace of AI advancement, tomorrow's general-purpose models may well outperform your carefully tuned older models with no customisation at all. Migrating fine-tuned weights between different model architectures is particularly challenging and requires a well-curated evaluation corpus. While LoRA is a valuable technique to have in your toolkit, it should only be deployed when the immediate business value clearly outweighs both the technical and opportunity costs.

Physical AI and robotics foundation models

Physical AI represents the convergence of foundation model capabilities with robotics and the physical world. Where traditional robotics relied on brittle, task-specific programming, robotics foundation models enable machines to generalise across tasks and adapt to novel situations. The investment signals are substantial: over $6 billion flowed into robotics companies in the first seven months of 2025, with individual raises including Figure AI ($1 billion), Physical Intelligence ($600 million), and Project Prometheus ($6.2 billion).

The technical breakthrough driving this shift is the emergence of Vision-Language-Action (VLA) models. These architectures extend the pattern established by vision-language models to include physical action outputs, enabling robots to interpret visual scenes and execute appropriate physical responses. NVIDIA's Isaac GR00T N1 represents the first open humanoid robot foundation model, using a dual-system architecture inspired by human cognition that separates deliberate planning from rapid reactive control. Google's Gemini Robotics models and the Genie 3 world model are advancing similar capabilities.

World Foundation Models (WFMs) complement robotics foundation models by enabling synthetic data generation and simulation-based training. NVIDIA Cosmos generates physically plausible synthetic environments that can train robots on scenarios too dangerous or rare to capture in the real world. This simulation-first approach dramatically reduces the cost and risk of developing physical AI systems.

We've placed this in Assess because whilst the technology is advancing rapidly, production deployments remain concentrated in well-resourced organisations with significant robotics expertise. The gap between research demonstrations and reliable industrial deployment is substantial. Hardware costs have decreased significantly, with capable platforms now available in the $2,000-$20,000 range compared to $75,000+ three years ago, but the integration challenges of perception and control in unstructured environments remain formidable. Organisations with physical AI ambitions should be actively experimenting and building capability, but should approach production timelines with appropriate caution.

See also: Digital twin platforms, World models

Hold

These techniques are not recommended for new projects due to better alternatives or limited long-term viability. While some may still have niche applications, they generally represent approaches that have been superseded by more effective solutions.

Word2Vec & GloVe

We've placed both GloVe (Global Vectors for Word Representation) and Word2Vec (Word to Vector) in the Hold ring of our techniques quadrant. While these word embedding techniques were groundbreaking when introduced and served as fundamental building blocks for many NLP applications, they have been largely superseded by more advanced approaches.

These older embedding techniques, though computationally efficient, lack the contextual understanding that modern transformer-based models provide. Modern large language models and contextual embeddings such as BERT produce more nuanced representations that capture word meaning based on surrounding context, rather than the static embeddings that GloVe and Word2Vec generate. For new projects, we recommend exploring more recent embedding techniques (see "BERT Variants" in our Adopt ring) unless you have very specific constraints around computational resources or model size that make these older approaches necessary.

t-SNE

We've placed t-SNE (t-distributed Stochastic Neighbor Embedding) in the Hold ring of our techniques quadrant. While t-SNE was groundbreaking when introduced for visualising high-dimensional data in lower dimensions, particularly for understanding the internal representations of neural networks, we're seeing its limitations become more apparent in modern AI workflows.

The core issue is that t-SNE can be misleading when interpreting AI model behaviour, as it prioritises preserving local structure at the expense of global relationships. This can lead teams to draw incorrect conclusions about their models' decision boundaries and feature representations. We're increasingly recommending alternatives such as UMAP (Uniform Manifold Approximation and Projection), which better preserves both local and global structure while offering superior computational performance. For projects requiring dimensionality reduction and visualisation of AI model internals, we suggest exploring these newer techniques rather than defaulting to t-SNE.

Zero-shot prompting

Zero-shot prompting, the practice of asking Large Language Models to perform tasks without examples or training, has been a quick way to get started with AI. However, we strongly recommend against using zero-shot prompts in production without appropriate guardrails and safety measures. We've heard of multiple incidents where unprotected prompts led to harmful or inappropriate outputs, potentially exposing organisations to significant risks.

Our view is that zero-shot prompting should always be combined with input validation and output filtering. While it can be valuable for prototyping and exploration, moving to few-shot prompting or fine-tuning with careful guardrails is a more robust approach for production systems. The current placement in Hold reflects our concern about organisations rushing to deploy unsafe prompt patterns rather than taking the time to implement proper controls.

Chain of thought (CoT)

Chain of Thought (CoT) has moved to our Hold ring. While CoT was a genuinely useful technique when it emerged, recent research from Wharton's Generative AI Labs demonstrates diminishing returns: gains are rarely worth the time cost, and for reasoning models such as o1 and o3, CoT prompting can actually decrease performance since step-by-step reasoning is already internalised at the architecture level.

For non-reasoning models, CoT still shows modest benefits on mathematical and symbolic reasoning tasks. However, these are precisely the domains where better alternatives are emerging. Dedicated reasoning models handle these tasks natively, while neurosymbolic architectures offer more reliable solutions by coupling LLMs with explicit symbolic reasoning engines rather than prompting models to simulate reasoning. CoT's remaining niche is being squeezed from both directions.

The frontier of prompt engineering has moved beyond eliciting reasoning to structuring problems effectively. Frameworks such as the 5 Whys and inversion ("what would guarantee failure?") now offer more value than CoT prompting. Resources such as taches-cc-resources catalogue these evolved approaches, focusing on problem framing and workflow orchestration rather than reasoning elicitation.

We're not suggesting that step-by-step reasoning is unimportant; quite the opposite. It's now so fundamental that it's handled by the models and architectures rather than the prompts.

See also: Neurosymbolic AI

AI pull request review

AI's code review capabilities have improved substantially. Developers who are accomplished at multi-turn conversations with AI can now get valuable feedback across the full spectrum: from syntax issues and style violations through architectural patterns to subtle runtime concerns such as race conditions. The days of AI only catching surface-level issues are behind us.

Yet we've kept AI Pull Request Review in Hold, and the reason is organisational rather than technical. PR review isn't just about finding errors; it's a vital knowledge-sharing mechanism where senior developers mentor juniors and the team maintains situational awareness of how the codebase is evolving. Teams who delegate review to AI often see a decline in collective code ownership and shared understanding.

The deeper question is how teams remain informed as AI handles more of the development workflow. Pull requests remain an excellent forum for this, perhaps more important than ever. We recommend using AI as a first-pass reviewer to catch issues before human review, but preserving the human review step as a deliberate practice for team alignment and knowledge transfer.

Tools

Software that enhances AI development workflows without being embedded in your application code: IDE extensions, CLI utilities, testing frameworks and observability solutions.

Adopt

44. Software engineering copilots

45. Provider-agnostic LLM facades

46. Notebooks

Trial

47. MLflow

48. Vector databases

49. Local model execution environments

50. LLM observability tools

Assess

51. AI application bootstrappers

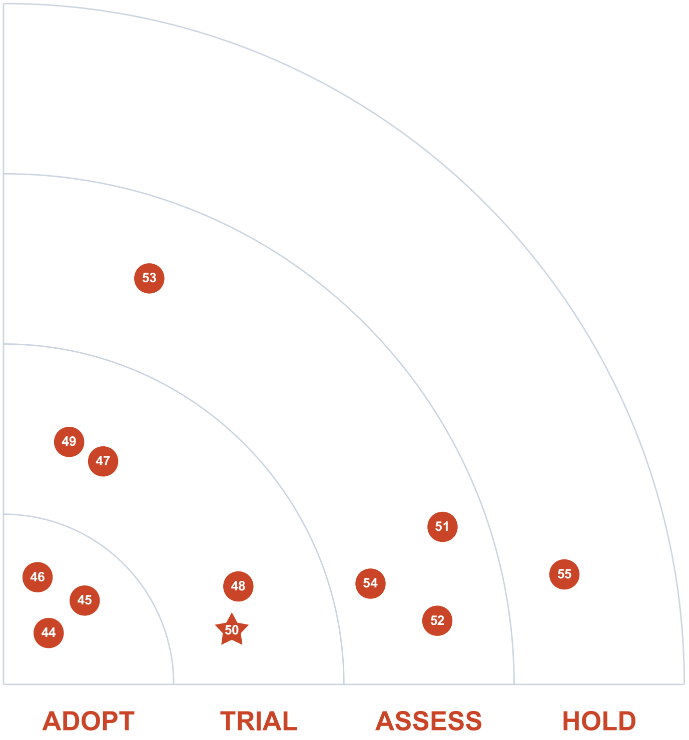
52. Visual computer use agents

53. Lakera

54. Structured output libraries

Hold

55. Conversational data analysis



Adopt

These tools represent mature, well-supported technologies that are ready for production use. They offer excellent productivity gains and proven track records in real-world development workflows.

Software engineering copilots

AI-augmented development represents the most significant shift in software engineering since the introduction of compilers. This transition is permanent, and teams that are not actively building capability in this area are already falling behind. We've placed Software Engineering Copilots firmly in the Adopt ring.

The tooling landscape offers two broad categories. Model-agnostic interfaces let teams choose and switch between LLM providers: OpenCode stands out for its polished terminal experience and breadth of integration, supporting 75+ model providers including local models for privacy-sensitive teams. Cursor, Windsurf and Zed are standalone editors supporting multiple models. CLI tools such as Aider and Cline work with various providers, Warp reimagines the terminal with AI-enhanced command suggestions, and Cody focuses on enterprise-scale codebase understanding. Provider-specific tools such as Claude Code, Gemini CLI and OpenAI Codex are optimised for their respective models. Claude Code has become extremely popular, reflecting the underlying model's capability; its interface has inspired several imitations. GitHub Copilot and Tabnine offer traditional IDE integrations with their own model stacks.

Two distinct approaches have emerged: free-form "vibe coding" and structured development methodologies. Kiro exemplifies this choice by offering both: a conversational coding mode for rapid iteration and a dedicated specs mode where AI assists developers in drafting requirements and design decisions through specification files before code generation. Traycer similarly emphasises upfront planning for complex tasks. Cursor enables teams to codify standards through .cursorrules, embedding architectural patterns and guidelines directly into AI assistance.

Senior engineers derive the greatest value, leveraging AI for routine tasks whilst maintaining quality oversight. Junior developers often struggle to evaluate AI suggestions, occasionally accepting flawed implementations or overlooking edge cases. This points to a clear organisational priority: intentional training around effective AI collaboration. Success correlates with careful workflow integration and a "trust but verify" mindset. The gap between teams that embrace these tools effectively and those that don't will only widen as the tooling continues to improve.

Provider-agnostic LLM facades

The LLM landscape evolves rapidly, making today's optimal choice potentially outdated within months. We recommend implementing a facade pattern between your application and LLM providers, rather than building directly against specific APIs. This approach reduces vendor lock-in and enables easier testing of alternative models as they emerge. When considering whether to write your own code, be sure to consider tools such as the lightweight AISuite, Simon Willison's LLM library and CLI tool, or heavyweight alternatives such as LangChain and LlamaIndex.

This recommendation reflects our team's experience seeing projects hampered by tight coupling to specific LLM providers, and the subsequent maintenance burden when transitioning to newer, more capable models.

Notebooks

We've placed Notebooks in the Adopt ring because they have become the de facto standard for data science and machine learning experimentation and prototyping. The interactive nature of notebooks, combining code execution with rich text explanations and visualisations, makes them particularly valuable for AI/ML workflows where iterative exploration and clear documentation of model development are essential.

Widespread adoption across both industry and academia, plus an extensive plugin ecosystem and integration with popular AI frameworks, demonstrates their maturity as a method of interacting with code. We especially value how notebooks facilitate collaboration between technical and non-technical team members, as they can serve as living documents that combine business requirements and technical implementation in a single, shareable format.

Jupyter notebooks are the most widely used, supporting multiple languages including Python and Julia. The cloud platforms provide their own implementations: Google Colab, AWS Sagemaker Notebooks, Azure Notebooks, Databricks Notebooks. And there are language specific notebooks, such as Pluto.jl for Julia, Clerk for Clojure, Polynote for Scala.

Trial

These tools show promising potential with growing adoption and active development. While they may not yet have the same maturity as Adopt tools, they offer innovative approaches and capabilities that make them worth exploring for forward-thinking teams.

MLflow

We have placed MLFlow in the Trial ring due to its potential as a lightweight and modular option for teams seeking to manage the machine learning lifecycle. Its open-source nature makes it an attractive alternative to the more monolithic cloud-based MLOps platforms provided by vendors such as AWS, Microsoft and Google. A key advantage of MLFlow is its ability to avoid vendor lock-in, offering teams the flexibility to maintain control of their infrastructure and adapt workflows as their needs evolve.

That said, realising the benefits of MLFlow requires teams to have a certain level of technical expertise to configure and integrate it into their existing systems effectively. Unlike cloud-native behemoths such as SageMaker or Vertex AI, MLFlow does not provide an all-in-one, plug-and-play experience. Instead, it offers modular components that must be tailored to specific use cases. We recommend assessing MLFlow if your organisation values flexibility, has the technical proficiency to manage integrations, and prefers avoiding dependency on proprietary platforms early in your MLOps journey.

Vector databases

Vector databases have emerged as specialised tools for managing the high-dimensional data representations (embeddings) required by AI models. They enable efficient similarity search across text and images. Prominent solutions include Pinecone, Qdrant, Milvus and Weaviate.

We've generally placed vector databases in the Trial ring, as they have proven valuable for specific use cases such as semantic search and recommendation systems. However, their adoption should be carefully evaluated based on individual requirements. Traditional databases may be sufficient for simpler operations and avoid the data consistency challenges of keeping embeddings synchronized with underlying content changes across databases. Alternative approaches, such as Timescale's PGAI vectorizer, bring vector embedding search directly into the Postgres database, ensuring embeddings remain synchronised with underlying content changes.

If a vector database is required for your use case, the choice of provider often depends on factors such as scale requirements and whether a managed or self-hosted solution is preferred. Pinecone leads in production readiness but comes with the costs of a managed service, while open-source alternatives such as Qdrant and Milvus offer greater control but demand more operational expertise.

For teams prioritising rapid prototyping and developer experience, Chroma has emerged as a popular choice. Its Python-first approach, minimal configuration requirements, and intuitive API make it accessible for developers without extensive database expertise. A 2025 Rust rewrite delivered significant performance improvements, and a new cloud offering extends its reach. However, Chroma remains best suited for prototyping and small-to-medium scale applications rather than enterprise workloads requiring SLAs and role-based access control.

LanceDB takes a different approach as an embedded vector database, similar in philosophy to SQLite. Rather than running a separate server, LanceDB operates as a library within your application, using Apache Arrow's columnar format to query vectors directly from disk at near-memory speeds. This makes it particularly compelling for local AI assistants, edge deployments, and scenarios where data must remain on-device. LanceDB is the only embedded vector database option in the Node.js ecosystem. The trade-off is that embedded architectures have inherent limits for high-concurrency workloads and can suffer cold-start latency in serverless environments.

Local model execution environments

Tools such as Ollama, LM Studio and AnythingLLM provide accessible ways to run open weight models on local hardware. These environments enable rapid experimentation with open weight models from providers including Meta (Llama), Mistral, DeepSeek, Alibaba (Qwen), and OpenAI (gpt-oss) without API costs or sending data to external services. Many now support advanced capabilities including web search, tool calling via Model Context Protocol (MCP), and connections to commercial APIs for hybrid workflows.

These tools serve various evaluation needs: developers testing AI features during development, teams comparing model responses for specific use cases, and organisations exploring AI capabilities with sensitive data that cannot leave their infrastructure. The range spans from command-line interfaces such as Ollama to graphical applications such as LM Studio, accommodating different technical backgrounds and preferences.

We've placed these in Trial as they offer a valuable alternative approach to model evaluation alongside cloud-based testing. They're particularly useful for privacy-sensitive prototyping and scenarios where extensive experimentation would be cost-prohibitive via APIs. Teams should consider these tools as one option among many for model evaluation, weighing their benefits against the overhead of local setup and maintenance.

LLM observability tools

The increasing complexity of agentic systems has created a need for development-time observability that goes beyond traditional production monitoring. When LLM applications involved simple API calls, understanding system behaviour was straightforward. Modern agentic builds involve multi-step reasoning, tool orchestration, RAG retrieval and chains of LLM calls where a single user request might trigger dozens of internal operations. Debugging why an agent produced an unexpected result requires visibility into every step of that chain.

We've created this section as distinct from Production AI monitoring platforms, which focus on drift detection and performance degradation in deployed systems. LLM observability tools address a different need: understanding what happened inside your application during development and debugging. As agentic architectures become more prevalent, this visibility becomes essential rather than optional.

Phoenix, from Arize AI, has emerged as a leading open-source option in this space. Built on OpenTelemetry, it avoids vendor lock-in whilst providing tracing and evaluation capabilities. Phoenix offers auto-instrumentation for popular frameworks including LangChain, LlamaIndex and DSPy, as well as direct integrations with OpenAI, Anthropic and AWS Bedrock. Teams can self-host for free or use Arize's cloud offering.

Langfuse is the most popular fully open-source alternative, available under MIT licence with no restrictions on self-hosting. It combines tracing and evaluation capabilities with strong support for multi-turn conversations. Langfuse integrates well with existing workflows and offers a generous free cloud tier for teams not ready to self-host.

For teams already committed to the LangChain ecosystem, LangSmith provides native integration that understands LangChain's internals and surfaces them in debugging views designed for that framework. Helicone takes a different approach as a lightweight proxy: route your API calls through Helicone's endpoint and gain observability without SDK changes, whilst also benefiting from gateway features such as caching and rate limiting.

Assess

These tools represent emerging or specialized technologies that may be worth considering for specific use cases. While they offer interesting capabilities, they require careful evaluation due to limited adoption or uncertain long-term viability.

AI application bootstrappers

AI application bootstrappers generate complete applications from prompts or designs. The market has matured rapidly, with Lovable (formerly GPT Engineer) emerging as a leader alongside established tools such as V0, Bolt.new and Replit Agent. Google entered the space in 2025 with Firebase Studio. These tools can dramatically accelerate the creation of demos and prototypes, taking projects from concept to working application in hours rather than days.

The capabilities are improving at a remarkable pace. Lovable's visual editor now allows Figma-like manipulation with automatic code updates. V0 excels at generating production-ready React components. Bolt.new runs full-stack development entirely in the browser. Enterprise adoption is growing, with major companies using these tools for internal tooling and rapid prototyping.

However, we remain cautious about production use. Success with these tools still correlates strongly with existing software engineering expertise. Senior developers can effectively use them as accelerators, understanding how to refactor generated code and establish proper architectural boundaries. Teams without this expertise risk shipping code they cannot maintain, debug, or evolve. The gap between "working demo" and "production-ready system" remains substantial, and we're particularly concerned about organisations building on bootstrapped foundations without the capability to evaluate what they've built.

We're watching this space with great interest. The pace of improvement suggests these tools may move toward Trial in future radars. For now, we recommend them primarily for prototyping and proof-of-concept work, with clear separation from production codebases unless your team has the engineering depth to take full ownership of the generated code.

Visual computer use agents

AI agents that interact with computers through visual understanding, controlling screens via mouse clicks and keyboard inputs as a human would, have matured but remain risky. Claude Computer Use allows Claude to control desktops and browsers by seeing the screen and reasoning about interface elements. OpenAI Operator, powered by their Computer-Using Agent (CUA) model, focuses on web browser automation through a managed environment. Browser Use offers an open-source alternative with flexibility across multiple model providers.

Reliability for bounded tasks has improved significantly, with standard office workflows now seeing success rates in the high 80s. However, serious security concerns temper any enthusiasm. Prompt injection attacks, where malicious instructions hidden on web pages hijack agent behaviour, represent a systemic vulnerability across all visual browser agents. OpenAI has publicly acknowledged that this problem "may never be fully solved," and security researchers warn that these agents "don't yet deliver enough value to justify their current risk profile" given their access to sensitive data like email and payment information.

We've kept visual computer use in the Assess ring. For many automation needs, programmatic approaches via APIs and workflow automation platforms remain both more reliable and more secure than visual interaction. Visual computer use is best suited to isolated environments where the agent cannot access sensitive data or navigate to untrusted websites. Teams considering these tools should grant minimal permissions, avoid broad instructions like "do whatever is needed," and maintain human oversight for any high-stakes actions.

This section was previously titled "Agentic computer use".

Lakera

Lakera is an AI safety and robustness platform designed to detect and mitigate risks in machine learning systems. It provides mechanisms for testing and analysis to help developers identify weaknesses or vulnerabilities in AI/ML models prior to deployment. This makes it particularly appealing in contexts where reliability and safety are paramount, such as finance, healthcare, or any domain subject to compliance constraints.

We have placed Lakera in the Assess ring because while it addresses an important need for AI safety, the platform has several practical limitations that require careful evaluation. Currently, Lakera supports only text-based scanning, teams using multimodal AI systems with images, audio, or video will find gaps in coverage. Custom scanning capabilities for business-specific terms or PII detection rely on regex patterns rather than context-aware analysis, which can quickly hit limitations in complex scenarios.

Performance considerations vary significantly between deployment options. The SaaS offering may provide adequate performance for many use cases, but has text size limitations that require applications to handle chunking. Self-hosted deployments offer more control but require substantial GPU resources for acceptable performance. Additionally, Lakera's scanning is non-stateful, each prompt and response is scanned in isolation without awareness of the broader conversation context, and only 'user' and 'assistant' message types are recognised.

Given these constraints, Lakera may provide valuable safety assurance for straightforward text-based AI applications, but organisations should carefully assess whether its current capabilities align with their specific AI architectures and safety requirements. We recommend conducting thorough proof-of-concept testing that includes your specific modalities and performance expectations before determining if Lakera fits your use case.

Structured output libraries

We've placed structured output libraries in the Assess ring as increasingly important tooling for production AI applications that need reliable, typed responses from LLMs.

Libraries such as Instructor, Outlines and Marvin address a common challenge. LLMs naturally produce freeform text, but applications typically need structured data: JSON matching a schema or selections from valid options. These libraries constrain LLM outputs to match specified structures, either through clever prompting, logit manipulation, or grammar-based generation.

The practical value is significant. Instead of hoping an LLM produces valid JSON and writing brittle parsing code, developers can specify Pydantic models and receive guaranteed-valid objects. This reduces error handling complexity and makes LLM outputs composable with traditional software. For agentic systems, structured outputs are essential. Agents need to produce function calls and decision objects that downstream code can reliably process.

We've placed these in Assess rather than Trial because the space is rapidly evolving and best practices are still emerging. Instructor has gained significant traction for its simplicity and Pydantic integration, while Outlines offers more sophisticated constrained generation for teams needing fine-grained control. Teams should evaluate which approach matches their reliability requirements and performance constraints. The native structured output features increasingly offered by model providers (OpenAI's JSON mode, Anthropic's tool use) may reduce the need for external libraries in some scenarios.

Hold

These tools are not recommended for new projects due to better alternatives or limited long-term viability. While some may still have niche applications, they generally represent technologies that have been superseded by more effective solutions.

Conversational data analysis

Tools such as pandas-ai, tablegpt, promptql, and Julius enable natural language querying of databases and datasets, offering significant productivity benefits for knowledgeable data analysts. Modern database-specific Model Context Protocol (MCP) servers can provide substantial context to models, including schema understanding and data contents. Our experience with JUXT's own XTDB database revealed remarkable moments where models navigated complex table structures with apparent ease, demonstrating genuine potential for accelerating data analysis workflows.

For experienced analysts, these tools represent a meaningful productivity boost, rapidly converting natural language requests into draft queries that can be refined and optimised. However, our experience also reveals challenges: generated queries can be inefficient or occasionally incorrect despite appearing plausible. The technology sometimes struggles with nuanced requirements and may produce suboptimal approaches that experienced analysts would avoid. Uber's experience with their internal QueryGPT tool demonstrates both the potential and the complexity, highlighting the significant number of example queries and guardrails required to achieve reliable results.

We've placed conversational data analysis in the Hold ring not because the technology lacks value, but because successful deployment requires users capable of understanding and validating generated queries. These tools offer substantial benefits for data teams with appropriate expertise, but should be approached cautiously by those unable to review and debug AI-generated database queries.

For teams with strong analytical capabilities, these tools can meaningfully accelerate exploratory data analysis and routine query generation, treating AI output as sophisticated first drafts requiring expert review.

Platforms

Infrastructure and managed services that host and run AI workloads: cloud AI services, vector databases, model serving platforms and MLOps infrastructure.

Adopt

56. Foundation models
57. Weights & Biases
58. Temporal
59. Data pipeline orchestration tools
60. Cloud model hosting platforms

Trial

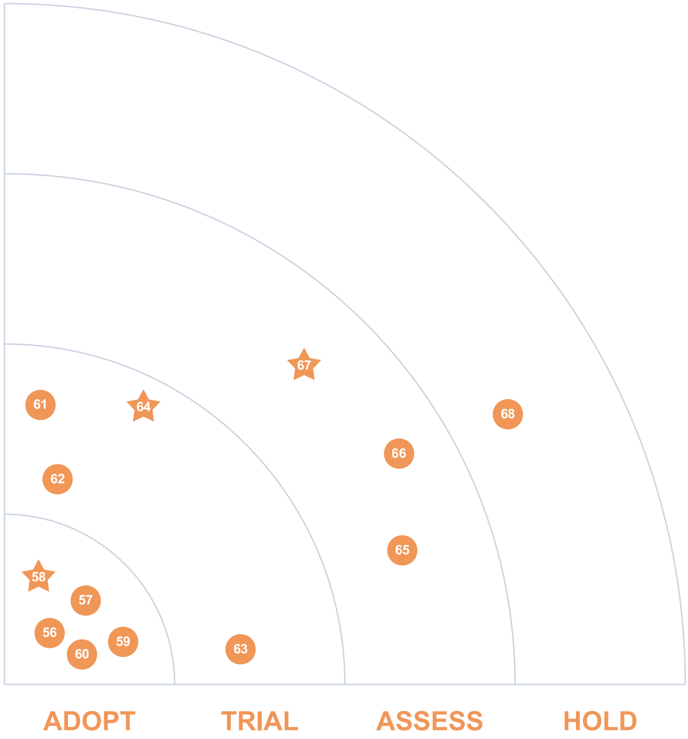
61. Production AI monitoring platforms
62. Open weight LLMs
63. AI-powered workflow automation platforms
64. Digital twin platforms

Assess

65. Galileo
66. Kubeflow
67. Process mining platforms

Hold

68. Building against vendor-specific APIs



Adopt

These platforms represent established, well-supported services that are ready for production use. They offer excellent reliability and proven track records in real-world AI deployments.

Foundation models

Foundation model providers continue to evolve at a rapid pace. Major players such as OpenAI, Anthropic, Google and Meta compete alongside emerging organisations including DeepSeek, Alibaba and IBM. While industry benchmarks help compare these models, they tell only part of the story: different models excel in different areas, and benchmark results should be viewed as indicative rather than definitive.

A clear trend has emerged in how providers differentiate their offerings across three distinct tiers: smaller, faster models (e.g., Claude Haiku, DeepSeek Coder, Qwen Turbo) optimised for speed and cost; larger, more capable models (e.g., Claude Sonnet, DeepSeek V3, Qwen Max) balancing capabilities with reasonable response times; and specialised reasoning models (e.g., Claude Sonnet Extended, OpenAI o1, DeepSeek R1) designed for complex problem-solving. These reasoning models consume significantly more tokens and command higher per-token costs, but demonstrate remarkable capabilities in solving challenging mathematical and coding tasks.

We believe foundation models have evolved sufficiently to warrant adoption for many business applications. When paired with appropriate infrastructure (few-shot prompting, guardrails, retrieval-augmented generation and evaluation frameworks), they offer compelling solutions to a wide range of problems. Our experience suggests there's no universal "best model". We recommend implementing your own benchmarking process focused on your specific use cases. When selecting a model, consider factors beyond raw performance, such as pricing, reliability, data privacy requirements, and whether on-premise deployment is needed. The recent emergence of high-quality open-source models with permissive licensing (such as DeepSeek's offerings) provides additional options for organisations with specific security or deployment requirements.

Key considerations

- **Performance & capabilities** (accuracy, speed, and domain-specific strengths)
- **Total cost of ownership** (API costs, compute resources, and integration)
- **Deployment options & technical requirements** (cloud, self-hosted, edge)
- **Data privacy & compliance** (regulatory, legal, and security implications)
- **Integration & lifecycle management** (context limitations, version control, updates)
- **Vendor stability & support** (roadmap alignment, documentation, community)

Foundation model providers feature comparison (January 2026)

Provider	Open Weights	Enterprise Focus	Reasoning Models	Edge Deployment	Long Context	Embedding API	Agentic Workflows	Model Selection Link
Alibaba	✓			✓	✓	✓		Models
Anthropic		✓	✓		✓		✓	Models
AWS		✓			✓			Models
Cohere	✓	✓	✓		✓	✓		Models
DeepSeek	✓		✓	✓				Models
Google		✓	✓		✓	✓	✓	Models
IBM	✓	✓	✓	✓	✓			Models
Meta	✓			✓				Models
MiniMax	✓			✓	✓		✓	Models
Mistral AI	✓	✓	✓	✓		✓		Models
OpenAI	✓	✓	✓	✓	✓	✓	✓	Models
Stability AI	✓			✓				Models
X	✓		✓		✓		✓	Models
Zhipu AI	✓	✓	✓		✓			Models

Feature definitions

- **Open Weights**: Models whose weights are publicly available for download and customisation
- **Enterprise Focus**: Strong emphasis on governance, security, and enterprise integration
- **Reasoning Models**: Specialised models for complex reasoning tasks such as mathematics or step-by-step problem solving
- **Edge Deployment**: Optimised for deployment on edge devices or resource-constrained environments
- **Long Context**: Support for context windows of 250K tokens or more
- **Embedding API**: Dedicated text embedding models and APIs for generating vector representations of text for semantic search and similarity tasks
- **Agentic Workflows**: Ability to autonomously plan and execute multi-step tasks using tools and external services. Goes beyond basic function calling to include complex workflow orchestration, error handling, dynamic planning based on intermediate results, and completing entire business processes without human intervention at each step

Weights & Biases

Weights & Biases is a platform designed for tracking and visualising machine learning experiments. In recent projects, we've observed that it provides a robust solution for managing machine learning workflows, particularly when dealing with complex models and large datasets. Its user-friendly interface and integration capabilities with popular machine learning libraries make it accessible for teams looking to improve their model development processes.

We've seen how systems such as Weights & Biases can catalyse positive cultural changes in ML teams. By making experiment tracking very light touch, requiring just a few lines of code, they remove the friction that sometimes prevents teams from maintaining good measurement practices. When tracking experiments becomes a natural part of the workflow rather than an extra burden, teams tend to measure more and make more data-driven decisions.

Collaboration features such as shared dashboards and reports amplify these benefits by making results and insights visible to the whole team. Rather than knowledge being siloed in individual notebooks or spreadsheets, experiments become shared assets that everyone can learn from. This visibility often leads to more discussion about results, faster knowledge sharing, and ultimately quicker iteration cycles as teams build upon each other's work rather than inadvertently duplicating efforts. However, it's important to note that tool adoption alone isn't enough, teams need to actively foster a culture that values measurement and experimentation for these benefits to fully materialise.

Temporal

We've placed Temporal in the Adopt ring as a workflow orchestration platform that provides durable execution for long-running, mission-critical processes. Although not AI-specific, Temporal has become increasingly relevant as organisations build production agentic systems that must survive failures and run reliably over extended periods.

The core value is durability. If a multi-step process fails halfway through, it resumes from exactly where it left off rather than restarting. This differs from infrastructure tools such as Kubernetes, which restart crashed containers but know nothing about application state. Kubernetes will restart your agent worker; Temporal will remember that your workflow was on step 5 of 10, waiting for human approval, with specific context variables intact. The two are complementary: Temporal typically runs on Kubernetes, with each handling failures at its respective layer.

Temporal's programming model treats workflows as ordinary code rather than configuration or visual diagrams. Developers write workflow logic in familiar languages (Go, Java, Python, TypeScript, .NET) with standard control flow and error handling. This makes complex orchestration easier to reason about and test than approaches based on state machines or YAML definitions.

For agentic systems specifically, Temporal addresses failure modes that many agent frameworks ignore: LLM calls timing out, tool invocations needing retry with backoff, human approvals taking days. Built-in retry policies and the ability to pause workflows indefinitely handle these scenarios cleanly. The platform also provides visibility into running workflows, making it possible to debug and monitor agent behaviour in production.

We recommend Temporal for organisations moving beyond prototype agents toward production deployments where reliability matters. The learning curve is moderate for teams familiar with distributed systems. For simpler use cases lighter-weight alternatives may suffice, but for mission-critical workflows Temporal provides a battle-tested foundation.

Data pipeline orchestration tools

Data pipeline orchestration has become essential infrastructure for organisations managing complex data workflows, particularly those supporting AI and machine learning initiatives. Whilst transformation tools such as dbt handle the "what" of data processing, orchestration platforms manage the "when," "how," and "monitoring" of entire pipelines. We've placed these tools in the Adopt ring because established organisations require systematic approaches to pipeline scheduling and failure recovery.

Apache Airflow represents the established approach, focusing on task-based workflows with broad integration support across cloud platforms. Its maturity and established ecosystem make it the de facto standard in many enterprises, though teams often find the learning curve steep. Prefect emphasises developer experience and dynamic workflow adaptation, allowing workflows to adapt to changing conditions with minimal code modification. Teams report faster development cycles, though fewer third-party integrations reflect the platform's relative youth.

Dagster takes an asset-centric approach where data assets become first-class citizens, providing built-in lineage tracking and data quality monitoring. This modern architecture includes comprehensive developer tooling and observability, though the conceptual shift from task-based thinking requires adjustment.

The choice between platforms typically depends on organisational context rather than technical superiority. Established enterprises with diverse toolchains often gravitate towards Airflow's ecosystem breadth, whilst teams prioritising developer velocity may prefer Prefect's flexibility. Organisations with complex data lineage requirements increasingly consider Dagster's asset-aware approach. We recommend evaluating these tools against your specific integration complexity, team expertise, and governance needs.

Cloud model hosting platforms

The model hosting landscape has evolved far beyond simple API access, with distinct platforms serving different organisational needs from rapid prototyping to enterprise production deployments. Each platform's approach to custom model deployment varies significantly, as organisations increasingly require hosting for their own fine-tuned models alongside foundation model access. We've placed these platforms in the Adopt ring because cloud-based model hosting has become the de facto approach for most AI deployments, reducing operational overhead.

Enterprise production environments often gravitate towards established cloud providers such as AWS Bedrock, Google Vertex AI, and Azure OpenAI Service. These platforms provide fine-tuning capabilities with enterprise security features and integration with existing cloud infrastructure. Azure's hub-and-spoke architecture (separating model training from deployment environments) and Google's "import custom model weights" feature automate parts of custom model deployment, though the processes often require cloud platform expertise and lengthy setup procedures.

Performance-critical applications are increasingly considering specialised providers such as Fireworks AI and Together AI, which focus specifically on inference optimisation and support deployment of custom fine-tuned models. These platforms offer API-based deployment workflows, with Together AI supporting trillion-parameter model training and Fireworks providing fine-tuning services. However, teams must evaluate whether simplified deployment compensates for reduced ecosystem integration compared to major cloud providers.

Development teams and startups often favour platforms such as Replicate, Modal, and Hugging Face Inference Endpoints, which emphasise deployment ease alongside flexible pricing. Hugging Face supports deployment of 60,000+ models with minimal configuration, whilst Replicate's Cog packaging system and Modal's Python-decorator approach reduce deployment steps. These platforms offer direct paths from trained model to production API, though enterprise governance features remain limited.

The choice between platforms reflects both organisational priorities and deployment complexity tolerance. Teams requiring sophisticated fine-tuning workflows with enterprise compliance often find major cloud providers necessary despite steeper learning curves. Performance-focused organisations benefit from specialised platforms that balance custom model support with optimisation capabilities. Development teams prioritising rapid iteration prefer platforms with simplified deployment processes, accepting more limited enterprise tooling.

Trial

These platforms show promising potential with growing adoption and active development. While they may not yet have the same maturity as Adopt platforms, they offer innovative approaches and capabilities that make them worth exploring for forward-thinking teams.

Production AI monitoring platforms

Whilst experiment tracking tools such as Weights & Biases and MLflow excel at managing the development lifecycle, a distinct category of platforms has emerged to monitor AI systems in production. These tools detect drift and unexpected behaviour in deployed models, issues that only surface when models encounter real-world data at scale. We've placed these platforms in the Trial ring as organisations continue establishing best practices for production AI monitoring.

Arize AI provides unified observability across traditional ML models and LLM applications, continuously tracking feature and embedding drift from training through to production. The platform helps catch production issues before customer impact, though careful configuration is needed to avoid alert fatigue. Evidently AI offers both an open-source library and cloud platform, with over 100 metrics covering data quality and drift monitoring. Its flexibility appeals to technical teams, though setup requires more effort than managed alternatives.

WhyLabs takes a privacy-preserving approach, monitoring through statistical profiles rather than raw data access. This enables massive scale monitoring whilst maintaining data security, particularly valuable for regulated industries. The platform claims superior drift detection accuracy, though teams must weigh privacy benefits against reduced debugging visibility.

Whilst there are many approaches to production AI monitoring, from custom metrics to manual spot checks, these platforms deserve consideration from teams hosting models in production. They integrate with existing SRE workflows through standard alerting channels (PagerDuty, Slack, email) and provide dashboards that fit alongside traditional application monitoring. The key benefit is proactive detection: organisations learn about performance degradation or prediction errors before customer impact, rather than discovering issues through support tickets. For teams already practising observability for their applications, adding AI-specific monitoring represents a natural extension of existing operational practices.

Open weight LLMs

2025 was the year when open weight LLMs (which are sometimes incorrectly referred to as 'open source') reached maturity, with some even surpassing flagship frontier models on certain tasks. Models such as MiniMax M2, Moonshot's Kimi K2, Zhipu's GLM-4, and DeepSeek V3 now compete directly with closed frontier models on coding and reasoning benchmarks. MiniMax M2 in particular has achieved the highest scores among open weight models on several intelligence indices whilst requiring only a fraction of the inference cost of comparable closed models. We've placed open weight LLMs in the Trial ring because they allow organisations to benefit from AI capabilities while maintaining control over their data and deployment. These models have demonstrated impressive performance, particularly in specialised domains when fine-tuned on specific tasks.

The key benefits include reduced operational costs compared to API-based services and full control over model deployment and customisation, along with the ability to run models in air-gapped environments where data privacy is paramount. However, we've kept them in Trial because organisations need considerable ML engineering expertise to deploy and maintain these models effectively, and the total cost of ownership isn't always lower than API-based alternatives when accounting for computational resources and engineering time.

For certain use cases, the simplicity of a pay-per-use API integration outweighs the benefits and greater control of hosting an open source LLM. Additionally, implementing appropriate security controls and data governance poses significant challenges.

AI-powered workflow automation platforms

Visual workflow automation platforms have become increasingly capable orchestrators of AI-powered business processes, allowing teams to build automated workflows through drag-and-drop interfaces rather than traditional coding. We've placed these platforms in the Trial ring because whilst they represent a maturing approach to democratising AI automation across organisations, the choice of platform depends heavily on specific technical and organisational requirements.

Prominent platforms in this space include Zapier, n8n, Microsoft Power Automate, and Make.com. Each serves different organisational needs and technical constraints. Zapier focuses on connecting SaaS applications with AI capabilities, positioning itself towards business users seeking rapid automation deployment. n8n distinguishes itself through flexibility for technical teams, offering self-hosting options, open-source licensing, and extensive customisation through HTTP nodes and JavaScript code injection. Microsoft Power Automate leverages native Office 365 integration and enterprise-grade governance features, whilst Make.com emphasises sophisticated visual workflow design with AI agent functionality.

These platforms attempt to bridge the gap between technical and business teams around AI automation. They allow organisations to prototype AI-enhanced workflows, connect disparate systems, and scale automation efforts without building custom integration layers. We've observed common use cases including lead qualification using LLM analysis, automated content generation and distribution, customer support ticket routing and responses, and data processing pipelines that incorporate AI models for classification or enrichment tasks.

When evaluating these platforms, teams should consider their organisation's technical capability, data sovereignty requirements, integration ecosystem needs, and long-term scalability plans. Self-hosted solutions such as n8n offer maximum control and customisation but require technical expertise, whilst SaaS offerings such as Zapier reduce operational overhead but may have cost implications at scale. Teams should also assess the platforms' capability for error recovery and debugging of AI-enhanced workflows, as AI components can fail in less predictable ways than traditional integrations.

Digital twin platforms

A digital twin is a virtual representation of a physical system that maintains bidirectional synchronisation with its real-world counterpart. Unlike traditional simulation or predictive analytics, which typically model a system at a point in time using historical data, digital twins continuously ingest live sensor data and update their state to reflect current reality. This enables organisations to ask "what if" questions against a model that represents the system as it exists now, not as it existed when the model was last trained.

The value emerges from this tight coupling between physical and virtual. Engineers can test changes in simulation before deploying them to production lines. Operators can diagnose problems by examining the digital twin's state without physical inspection. Planners can simulate the impact of new equipment or layout changes on existing workflows. In robotics specifically, digital twins enable training AI systems in simulation before exposing them to the costs and risks of physical deployment.

NVIDIA Omniverse has emerged as the dominant platform for industrial digital twins, providing a simulation environment built on OpenUSD (Universal Scene Description) that enables physically accurate rendering and real-time collaboration. Its Isaac Sim extension specifically targets robotics simulation. Major manufacturers including Foxconn, Toyota and Caterpillar are using Omniverse to design and simulate factory layouts and assembly lines. A 2025 Hexagon survey found that over 90% of enterprises deploying digital twins report positive ROI, though the magnitude varies significantly with implementation maturity.

We've placed digital twin platforms in Trial because whilst the technology is mature, successful deployment requires significant organisational investment beyond the platform itself. The challenge is rarely the simulation technology; it lies in data integration, maintaining synchronisation between physical and virtual systems, and building organisational capability to act on simulation insights. Teams that approach digital twins as purely technical projects often struggle to scale beyond proof-of-concept.

Digital twins are most relevant for organisations operating complex physical infrastructure: manufacturing plants, logistics networks, energy systems, building management, or robotics deployments. Industries with high costs of physical experimentation or downtime see the clearest benefits. Organisations earlier in their data maturity journey should ensure foundational sensor instrumentation and data pipelines are in place before investing in digital twin platforms.

See also: Physical AI and robotics foundation models

Assess

These platforms represent emerging or specialized services that may be worth considering for specific use cases. While they offer interesting capabilities, they require careful evaluation due to limited adoption or uncertain long-term viability.

Galileo

We've placed Galileo in the Assess ring of the Platforms radiant because it represents an interesting approach to evaluating and improving AI model performance. It deserves attention but requires careful consideration before being adopted more broadly.

Galileo offers a comprehensive platform spanning both development evaluation and production monitoring of AI systems. During development, it provides tools for measuring and refining model performance, with specialised capabilities for AI agent evaluation and comprehensive testing frameworks. In production, the platform offers real-time monitoring with low-latency guardrails and hallucination detection. Our committee has noted that teams using the platform report better insights into how their AI systems perform across different scenarios and edge cases, from initial development through to production deployment.

We recommend assessing this platform, particularly if your organisation is developing custom models or fine-tuning existing ones, as the insights it provides could significantly improve model quality. However, we've stopped short of recommending it for trial by all teams, as its value varies depending on your level of AI maturity and your specific use cases. Organisations with simpler AI implementations, or those primarily using out-of-the-box models, may find less immediate benefit. The platform is likely to offer the most value to organisations that are actively developing or fine-tuning models, or deploying AI in high-stakes environments where consistent performance is critical. Teams should also consider whether they have the technical resources required to act effectively on the insights the platform provides.

Kubeflow

We've placed Kubeflow in the Assess ring of our Platforms quadrant. This open-source machine learning platform, built on Kubernetes, offers a comprehensive solution for managing ML workflows, but it requires careful evaluation before widespread adoption.

Kubeflow is gaining traction among data science and MLOps teams looking to standardise their machine learning workflows. Its strength lies in combining Kubernetes' orchestration capabilities with ML-specific tools: Pipelines for workflow automation and KFServing for model deployment. This integrated approach helps bridge the gap between data scientists and operations teams, addressing one of the core challenges in operationalising ML models.

However, several factors keep Kubeflow in our Assess ring. First, implementing Kubeflow demands significant expertise in both Kubernetes and ML engineering, a specialised skill set that remains relatively uncommon. Second, while the platform is maturing, we've observed that many organisations struggle with its complexity during initial setup and ongoing maintenance. Teams often report a steep learning curve before realising tangible benefits.

Organisations with established ML practices and existing Kubernetes expertise should consider assessing Kubeflow, particularly if they're facing challenges with ML model deployment, experiment reproducibility or resource utilisation. The platform is especially suited to enterprises managing multiple ML models in production that require systematic oversight across their lifecycle. Smaller teams, or those earlier in their ML journey, may want to explore simpler alternatives first or consider managed options such as Vertex AI Pipelines, which abstract away some of the infrastructure complexity.

Process mining platforms

You cannot reliably automate processes that have not yet been optimised. Our experience suggests that 80% of a process usually flows as expected, but it's the remaining 20%, the exceptions and edge cases, that determine whether automation succeeds or fails. As agentic AI moves from experimentation to enterprise deployment, process mining is emerging as essential preparation. Enterprise software vendors are converging on the idea that process mining enables agentic AI by revealing process inefficiencies, violations, and their root causes; this helps to deduce where automation would bear the most fruit, and active monitoring reinforces ongoing improvement.

Process mining uses event logs from enterprise systems to assemble and analyse a digital twin of the process that helps to discover how processes actually execute versus the designed workflow. The related discipline of task mining records user activity at the desktop level, capturing the tacit knowledge of how workers handle exceptions and workarounds. Together, these capabilities map the reality that AI agents would need to replicate.

Our experience with the major platforms reveals clear differentiation regarding which tool fits which infrastructure:

Celonis, the market leader, boasts the most sophisticated analytics and the largest community, with the founder of the process mining methodology (Professor Wil van der Aalst) at its core. It requires significant investment but delivers the deepest insights for complex, multi-system processes. Best suited for mature, large-scale enterprises with dedicated process excellence teams and budget.

ABBYY Timeline offers well-integrated process and task mining in a more accessible package than Celonis. The point-and-click interface suits business users who want to identify bottlenecks without coding. A pragmatic middle ground for organisations wanting capable tooling including task mining without an enterprise-scale infrastructure commitment.

Platforms

QPR ProcessAnalyzer is a powerful tool which is also available in the Snowflake Marketplace. Originally focused on consulting and process improvement, QPR developed leading process modelling software before launching their highly configurable mining tool that put them in Gartner's Visionaries quadrant. The intuitive choice for companies with Snowflake at their core, as the initial setup takes less than five minutes.

UiPath Process Mining, where the integration between process discovery and automation execution is seamless, though this does create ecosystem lock-in. The natural choice for organisations already invested in UiPath's RPA ecosystem who want to bridge the gap between mining and acting.

Microsoft Power Automate Process Mining, while limited compared to specialist platforms, has integration with Power Automate and Copilot that lowers the barrier for organisations beginning their journey. It represents the path of least resistance for organisations preferring Microsoft products and users already familiar with Power BI who want to leverage their existing stack.

Fluxicon Disco is a favourite among process mining purists. Disco is a standalone desktop application known for its speed and simplicity, allowing users to import data and immediately visualise process maps without complex server installations. It is best suited for consultants and rapid proof of concept projects where the goal is quick, ad-hoc analysis rather than continuous, automated enterprise monitoring.

We recommend starting with process discovery in a contained domain rather than attempting an enterprise-wide rollout. This validates that insights genuinely inform automation design before scaling investment.

Hold

These platforms are not recommended for new projects due to better alternatives or limited long-term viability. While some may still have niche applications, they generally represent approaches that have been superseded by more effective solutions.

Building against vendor-specific APIs

We've placed "Building against vendor-specific APIs" in the Hold ring of the Platforms quadrant because tightly coupling your applications to vendor-specific LLM APIs poses significant business risks in this rapidly evolving landscape.

The foundation model ecosystem is changing at breakneck speed, with model capabilities, pricing and even entire companies shifting dramatically from month to month. Organisations that build directly against OpenAI, Anthropic or other proprietary APIs often find themselves locked in, facing painful migrations when a better or more cost-effective model emerges. We've seen teams invest substantial engineering effort into rewriting API integrations after discovering their chosen vendor has been outperformed or has significantly increased its pricing.

Instead, we recommend using abstraction libraries that provide a common interface to multiple LLM providers. Libraries such as AISuite or Simon Willison's LLM CLI let you switch between different models with minimal code changes, sometimes just a configuration update. These libraries handle the nuances of different vendor APIs, managing context windows, token limitations and provider-specific parameters behind a consistent interface. This approach preserves your flexibility to take advantage of new capabilities or improved pricing as the market evolves, while significantly reducing the engineering effort required to switch between models.

These abstractions do add some complexity and may occasionally limit access to vendor-specific features, but in our view, the protection against vendor lock-in far outweighs these drawbacks in most cases. As the foundation model market continues to consolidate, maintaining the flexibility to adapt quickly will be crucial for both cost management and staying competitive.

juxt.pro/ai-radar

The information in this document is provided as is and does not warrant the accuracy, completeness and fitness for a particular purpose. In no event shall Grid Dynamics Holdings, Inc be liable for any damages whatsoever arising from your reliance on any information contained in this document. Except as permitted under the Copyright Act no part of the document may be reproduced, transmitted, stored in a retrieval system, or translated into any human or computer language in any form by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise without the expressed permission of Grid Dynamics Holdings, Inc.