AI Agents and LangChain Ecosystem - Advanced Guide

Enhanced AI Agent Features

What are AI agents? Definition, examples, and types | Google CloudWhat is an AI agent?AI agents are software systems that use AI to pursue goals and complete tasks on behalf of users. They show reasoning, planning, and memory and have a level of autonomy to make decisions, learn, and adapt.Their capabilities are made possible in large part by the multimodal capacity of generative AI and AI foundation models. AI agents can process multimodal information like text, voice, video, audio, code, and more simultaneously; can converse, reason, learn, and make decisions. They can learn over time and facilitate transactions and business processes. Agents can work with other agents to coordinate and perform more complex workflows.Demo Vertex AIStay informedExplore 300+ real-world gen AI use cases from the world's leading organizationsRead the blogKey features of an AI agentAs explained above, while the key features of an AI agent are reasoning and acting (as described in ReAct Framework) more features have evolved over time.Reasoning: This core cognitive process involves using logic and available information to draw conclusions, make inferences, and solve problems. AI agents with strong reasoning capabilities can analyze data, identify patterns, and make informed decisions based on evidence and context.Acting: The ability to take action or perform tasks based on decisions, plans, or external input is crucial for AI agents to interact with their environment and achieve goals. This can include physical actions in the case of embodied AI, or digital actions like sending messages, updating data, or triggering other processes.Observing: Gathering information about the environment or situation through perception or sensing is essential for AI agents to understand their context and make informed decisions. This can involve various forms of perception, such as computer vision, natural language processing, or sensor data analysis.Planning: Developing a strategic plan to achieve goals is a key aspect of intelligent behavior. AI agents with planning capabilities can identify the necessary steps, evaluate potential actions, and choose the best course of action based on available information and desired outcomes. This often involves anticipating future states and considering potential obstacles.Collaborating: Working effectively with others, whether humans or other AI agents, to achieve a common goal is increasingly important in complex and dynamic environments. Collaboration requires communication, coordination, and the ability to understand and respect the perspectives of others.Self-refining: The capacity for self-improvement and adaptation is a hallmark of advanced AI systems. AI agents with self-refining capabilities can learn from experience, adjust their behavior based on feedback, and continuously enhance their performance and capabilities over time. This can involve machine learning techniques, optimization algorithms, or other forms of self-modification.What is the difference between AI agents, AI assistants, and bots?AI assistants are AI agents designed as applications or products to collaborate directly with users and perform tasks by understanding and responding to natural human language and inputs. They can reason and take action on the users' behalf with their

supervision.AI assistants are often embedded in the product being used. A key characteristic is the interaction between the assistant and user through the different steps of the task. The assistant responds to requests or prompts from the user, and can recommend actions but decision-making is done by the user.AI agentAI assistantBotPurposeAutonomously and proactively perform tasksAssisting users with tasksAutomating simple tasks or conversationsCapabilitiesCan perform complex, multi-step actions; learns and adapts; can make decisions independentlyResponds to requests or prompts; provides information and completes simple tasks; can recommend actions but the user makes decisionsFollows pre-defined rules; limited learning; basic interactionsInteractionProactive; goal-orientedReactive; responds to user requestsReactive; responds to triggers or commandsAI agentAI assistantBotPurposeAutonomously and proactively perform tasksAssisting users with tasksAutomating simple tasks or conversationsCapabilitiesCan perform complex, multi-step actions; learns and adapts; can make decisions independentlyResponds to requests or prompts; provides information and completes simple tasks; can recommend actions but the user makes decisionsFollows pre-defined rules; limited learning; basic interactionsInteractionProactive; goal-orientedReactive; responds to user requestsReactive; responds to triggers or commandsKey differencesAutonomy: AI agents have the highest degree of autonomy, able to operate and make decisions independently to achieve a goal. AI assistants are less autonomous, requiring user input and direction. Bots are the least autonomous, typically following pre-programmed rules.Complexity: AI agents are designed to handle complex tasks and workflows, while AI assistants and bots are better suited for simpler tasks and interactions.Learning: AI agents often employ machine learning to adapt and improve their performance over time. AI assistants may have some learning capabilities, while bots typically have limited or no learning.How do AI agents work?Every agent defines its role, personality, and communication style, including specific instructions and descriptions of available tools. Persona: A well defined persona allows an agent to maintain a consistent character and behave in a manner appropriate to its assigned role, evolving as the agent gains experience and interacts with its environment.Memory: The agent is equipped in general with short term, long term, consensus, and episodic memory. Short term memory for immediate interactions, long-term memory for historical data and conversations, episodic memory for past interactions, and consensus memory for shared information among agents. The agent can maintain context, learn from experiences, and improve performance by recalling past interactions and adapting to new situations.Tools: Tools are functions or external resources that an agent can utilize to interact with its environment and enhance its capabilities. They allow agents to perform complex tasks by accessing information, manipulating data, or controlling external systems, and can be categorized based on their user interface, including physical, graphical, and program-based interfaces. Tool learning involves teaching agents how to effectively use these tools by understanding their functionalities and the context in which they should be applied.Model: Large language models (LLMs) serve as the foundation for building AI agents, providing them with the ability to understand, reason, and act. LLMs act as the "brain" of an agent, enabling them to process and generate language, while other components facilitate reason and action.What are the types

of agents in AI?AI agents can be categorized in various ways based on their capabilities, roles, and environments. Here are some key categories of agents:There are different definitions of agent types and agent categories. Based on interactionOne way to categorize agents is by how they interact with users. Some agents engage in direct conversation, while others operate in the background, performing tasks without direct user input: Interactive partners (also known as, surface agents) - Assisting us with tasks like customer service, healthcare, education, and scientific discovery, providing personalized and intelligent support. Conversational agents include Q&A, chit chat, and world knowledge interactions with humans. They are generally user query triggered and fulfill user queries or transactions.Autonomous background processes (also known as, background agents) - Working behind the scenes to automate routine tasks, analyze data for insights, optimize processes for efficiency, and proactively identify and address potential issues. They include workflow agents. They have limited or no human interaction and are generally driven by events and fulfill queued tasks or chains of tasks.Based on number of agentsSingle agent: Operate independently to achieve a specific goal. They utilize external tools and resources to accomplish tasks, enhancing their functional capabilities in diverse environments. They are best suited for well defined tasks that do not require collaboration with other AI agents. Can only handle one foundation model for its processing.Multi-agent: Multiple AI agents that collaborate or compete to achieve a common objective or individual goals. These systems leverage the diverse capabilities and roles of individual agents to tackle complex tasks. Multi-agent systems can simulate human behaviors, such as interpersonal communication, in interactive scenarios. Each agent can have different foundation models that best fit their needs.Benefits of using AI agentsAI agents can enhance the capabilities of language models by providing autonomy, task automation, and the ability to interact with the real world through tools and embodiment.Expand allEfficiency and productivityIncreased output: Agents divide tasks like specialized workers, getting more done overall.Simultaneous execution: Agents can work on different things at the same time without getting in each other's way.Automation: Agents take care of repetitive tasks, freeing up humans for more creative work.Improved decision-makingCollaboration: Agents work together, debate ideas, and learn from each other, leading to better decisions.Adaptability: Agents can adjust their plans and strategies as situations change.Robust reasoning: Through discussion and feedback, agents can refine their reasoning and avoid errors.Enhanced capabilitiesComplex problem-solving: Agents can tackle challenging real-world problems by combining their strengths.Natural language communication: Agents can understand and use human language to interact with people and each other.Tool use: Agents can interact with the external world by using tools and accessing information.Learning and self-improvement: Agents learn from their experiences and get better over time.Social interaction and simulationRealistic simulations: Agents can model human-like social behaviors, such as forming relationships and sharing information.Emergent behavior: Complex social interactions can arise organically from the interactions of individual agents.Challenges with using AI agentsWhile AI agents offer many benefits, there are also some challenges associated with their use:Tasks requiring deep empathy / emotional intelligence or requiring complex human interaction and social dynamics - AI agents can struggle with nuanced human

emotions. Tasks like therapy, social work, or conflict resolution require a level of emotional understanding and empathy that AI currently lacks. They may falter in complex social situations that require understanding unspoken cues.Situations with high ethical stakes - AI agents can make decisions based on data, but they lack the moral compass and judgment needed for ethically complex situations. This includes areas like law enforcement, healthcare (diagnosis and treatment), and judicial decision-making. Domains with unpredictable physical environments - AI agents can struggle in highly dynamic and unpredictable physical environments where real-time adaptation and complex motor skills are essential. This includes tasks like surgery, certain types of construction work, and disaster response.Resource-intensive applications - Developing and deploying sophisticated AI agents can be computationally expensive and require significant resources, potentially making them unsuitable for smaller projects or organizations with limited budgets.Use cases for AI agentsOrganizations have been deploying agents to address a variety use cases, which we group into six key broader categories:Customer agentsCustomer agents deliver personalized customer experiences by understanding customer needs, answering questions, resolving customer issues, or recommending the right products and services. They work seamlessly across multiple channels including the web, mobile, or point of sale, and can be integrated into product experiences with voice or video. Employee agentsEmployee agents boost productivity by streamlining processes, managing repetitive tasks, answering employee questions, as well as editing and translating critical content and communications. Creative agentsCreative agents supercharge the design and creative process by generating content, images, and ideas, assisting with design, writing, personalization, and campaigns. Data agentsData agents are built for complex data analysis. They have the potential to find and act on meaningful insights from data, all while ensuring the factual integrity of their results. Code agentsCode agents accelerate software development with AI-enabled code generation and coding assistance, and to ramp up on new languages and code bases. Many organizations are seeing significant gains in productivity, leading to faster deployment and cleaner, clearer code. Security agentsSecurity agents strengthen security posture by mitigating attacks or increasing the speed of investigations. They can oversee security across various surfaces and stages of the security life cycle: prevention, detection, and response. Google Cloud and AI agentsGoogle Cloud provides a portfolio of products and solutions in the AI agent space. These include integrated AI assistants, pre-built AI agents, AI applications, and a platform of agent and developer tools to build custom AI agents.Vertex AI Agent BuilderCreate AI agents and applications using natural language or a code-first approach. Easily ground your agents or apps in enterprise data with a range of options. Conversational Agents and DialogflowBuild hybrid conversational agents with both deterministic and generative AI functionality. Google AgentspaceConnects your work apps to Google-quality multimodal search and the power of AI agents. Vertex AI Agent EngineFully managed runtime to deploy and manage agents with a simple SDK and APIs, wrap any agent in any python based framework and deploy it quickly. Vertex AI Agent Garden (Github)Curated collection of pre-built agent samples, solutions, tools, and frameworks to accelerate the development and deployment of AI agents.Agent Development Kit (ADK)Open-source Python SDK to build sophisticated multi-agent systems with

orchestration, memory, and developer tools. Additional resourcesContinue learning about AI agents with additional resources.Google ADK on GithubGoogle Agents White Paper (via Kaggle)Google Agents Companion White Paper (via Kaggle)Skillsboost Advanced Generative AI for Developers Learning Take the next stepStart building on Google Cloud with $300 in free credits and 20+ always free products.Get started for freeNeed help getting started?Contact salesWork with a trusted partnerFind a partnerContinue browsingSee all products Core AI Agent Architecture: - Perception Layer: Sensors and data input mechanisms - Reasoning Engine: Decision-making using LLMs and ML algorithms - Action Layer: Execution capabilities and tool integration - Memory System: Short-term and long-term information storage Advanced Capabilities: - Multi-step reasoning and planning - Tool use and API integration - Collaborative multi-agent workflows - Adaptive learning from environment feedback - Context-aware decision making Real-world Applications: - Customer service automation with contextual understanding - Supply chain optimization and demand forecasting - Content creation and curation workflows - Code generation and software development assistance - Research and data analysis automation

Advanced LangChain Framework

Introduction to LangChain - GeeksforGeeks Data Science IBM CertificationData ScienceData Science ProjectsData AnalysisData VisualizationMachine LearningML ProjectsDeep LearningNLPComputer VisionArtificial Intelligence Sign In Open In App Explore GfG CoursesShare Your ExperiencesIntroduction to LangsmithIntroduction to LangsmithBuild Chatbot Webapp with LangChainiNLTK: Natural Language Toolkit for Indic Languages in PythonDevelop an LLM Application using OpenaiSimple Steps to Learn Any Programming Language in 2025Nation SkillUp Explore Introduction to LangChain Last Updated : 06 Feb, 2025 Comments Improve Suggest changes Like Article Like Report LangChain is an open-source framework designed to simplify the creation of applications using large language models (LLMs). It provides a standard interface for chains, many integrations with other tools, and end-to-end chains for common applications. LangChain allows AI developers to develop applications based on the combined Large Language Models (such as GPT-4) with external sources of computation and data. This framework comes with a package for both Python and JavaScript.Why is LangChain Important?LangChain helps manage complex workflows, making it easier to integrate LLMs into various applications like chatbots and document analysis. Key benefits include:Modular Workflow: Simplifies chaining LLMs together for reusable and efficient workflows.Prompt Management: Offers tools for effective prompt engineering and memory handling.Ease of Integration: Streamlines the process of building LLM-powered applications.Key Components of LangChain1. ChainsChains define sequences of actions, where each step can involve querying an LLM, manipulating data, or interacting with external tools. There are two types:Simple Chains: A single LLM invocation.Multi-step Chains: Multiple LLMs or actions combined, where each step can take the output from the previous step.2. Prompt ManagementLangChain facilitates managing and customizing prompts passed to the LLM. Developers can use PromptTemplates to define how inputs and outputs are formatted before being passed to the model. It also simplifies tasks like handling dynamic variables and prompt engineering, making it easier to control

the LLM's behavior.3. AgentsAgents are autonomous systems within LangChain that take actions based on input data. They can call external APIs or query databases dynamically, making decisions based on the situation. These agents leverage LLMs for decision-making, allowing them to respond intelligently to changing input.4. Vector DatabaseLangChain integrates with a vector database, which is used to store and search high-dimensional vector representations of data. This is important for performing similarity searches, where the LLM converts a query into a vector and compares it against the vectors in the database to retrieve relevant information.Vector database plays a key role in tasks like document retrieval, knowledge base integration, or context-based search, providing the model with dynamic, real-time data to enhance responses.5. ModelsLangChain is model-agnostic, meaning it can integrate with different LLMs, such as OpenAI's GPT, Hugging Face models, DeepSeek R1, and more. This flexibility allows developers to choose the best model for their use case while benefiting from LangChain's architecture.6. Memory ManagementLangChain supports memory management, allowing the LLM to "remember" context from previous interactions. This is especially useful for creating conversational agents that need context across multiple inputs. The memory allows the model to handle sequential conversations, keeping track of prior exchanges to ensure the system responds appropriately.How LangChain Works?LangChain follows a structured pipeline that integrates user queries, data retrieval and response generation into seamless workflow. LangChain Pipeline1. User QueryThe process begins when a user submits a query or request. For example, a user might ask, "What's the weather like today?" This query serves as the input to the LangChain pipeline.2. Vector Representation & Similarity SearchOnce the query is received, LangChain converts it into a vector representation using embeddings. This vector captures the semantic meaning of the query. The vector is then used to perform a similarity search in a vector database. The goal is to find the most relevant information or context stored in the database that matches the user's query.3. Fetching Relevant InformationBased on the similarity search, LangChain retrieves the most relevant data or context from the database. This step ensures that the language model has access to accurate and contextually appropriate information to generate a meaningful response.4. Generating a ResponseThe retrieved information is passed to the language model (e.g., OpenAI's GPT, Anthropic's Claude, or others). The LLM processes the input and generates a response or takes an action based on the provided data. For example, if the query is about the weather, the LLM might generate a response like, "Today's weather is sunny with a high of 75F."The formatted response is returned to the user as the final output. The user receives a clear, accurate, and contextually relevant answer to their query.Getting Started with LangChainTo get started with LangChain, you'll need to install the required libraries and set up a basic environment.Step 1: Install LangChainTo install LangChain, use the following command:!pip install langchainStep 2: Install OpenAILangChain works with various Large Language Models (LLMs), and for this example, we'll be using OpenAI. To install OpenAI, run the following:!pip install openaiStep 3: Install Python-dotenvFor storing the OpenAI API key securely in an environment variable, we'll use the python-dotenv library. Install it by running:!pip install python-dotenvStep 4: Generate and Store Your API KeyYou need to generate your API key from the OpenAI platform by signing up and creating an account. To learn, how can we access the API key from Open AI

refer: What is ChatGPT API?Once you have the API key, create a .env file in your project directory and add your API key to it like this: Python OPENAI_KEY='your_api_key' Step 5: Set Up Your Python ScriptNext, create a new Python file named lang.py. In this file, you'll use LangChain to generate responses with OpenAI. Start by importing the necessary libraries: Python import os import openai import langchain from dotenv import load_dotenv # Load the API key from .env file load_dotenv() api_key = os.getenv("OPENAI_KEY", None) This code loads the environment variables from the .env file, where your OpenAI API key is stored.Building an Application using LangChainNow that the initial setup is complete, let's move on to building a simple application that generates responses. We'll start by asking the model a basic question: "Suggest me a skill that is in demand?"Step 1: Initialize the OpenAI ModelImport LangChain and initialize the OpenAI LLM (Large Language Model) using the OpenAI class: Python from langchain.llms import OpenAI # Initialize OpenAI LLM with a temperature of 0.9 for randomness llm = OpenAI(temperature=0.9, openai_api_key=api_key) In this case, the temperature=0.9 setting means the results will be more random and creative. If you want the responses to be more accurate and deterministic, you can lower the temperature to around 0.4.Step 2: Generate a ResponseNow that the model is initialized, you can generate a response by passing a simple prompt to it. In this case, we'll ask, "Suggest me a skill that is in demand?" Python response=llm.predict("Suggest me a skill that is in demand?") print(response) Output: One skill in demand right now is software/web development, which includes everything from coding to content management systems to web design. Other skills in demand include cloud computing, machine learning and artificial intelligence, digital marketing, cybersecurity, data analysis, and project management.This output is based on the language model's prediction, and with the temperature setting of 0.9, it provides a more creative and diverse set of skills that are currently in demand.Applications of LangChainLangChain is a powerful tool that can be used to build a wide range of LLM-powered applications. It is simple to use and has a large user and contributor community.Conversational Agents: Build chatbots and virtual assistants that can engage in meaningful, context-aware conversations with users.Document Summarization: Automatically generate summaries of long documents, making it easier for users to digest large amounts of information.Question Answering: Create systems that can answer questions based on a given context or a set of documents.Workflow Automation: Design workflows that involve multiple steps, such as data extraction, processing, and reporting, all powered by language models.Content Generation: Generate creative content, such as articles, stories, or marketing copy, with the help of language models.The LangChain framework is a great interface to develop interesting AI-powered applications and from personal assistants to prompt management as well as automating tasks. So, keep learning and keep developing powerful applications. Introduction to Langchain Comment More infoAdvertise with us Next Article Build Chatbot Webapp with LangChain N namaldesign Follow Improve Article Tags : Artificial Intelligence Python-Library Python-Miscellaneous python ChatGPT Prompts OpenAI API +2 More Practice Tags : python Like 264k+ interested Geeks Master Competitive Programming - Complete Beginner to Advanced Explore 4k+ interested Geeks GATE CSE 2027 Online Course [Live Classes] Explore 34k+ interested Geeks GATE CSE Rank Booster with Expert-Curated Questions Explore We use cookies to ensure you have the best browsing experience on our

website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy Got It ! Improvement Suggest changes Suggest Changes Help us improve. Share your suggestions to enhance the article. Contribute your expertise and make a difference in the GeeksforGeeks portal. Create Improvement Enhance the article with your expertise. Contribute to the GeeksforGeeks community and help create better learning resources for all. Suggest Changes min 4 words, max Words Limit:1000 Thank You! Your suggestions are valuable to us. What kind of Experience do you want to share? Interview Experiences Admission Experiences Career Journeys Work Experiences Campus Experiences Competitive Exam Experiences Advanced LangChain Components: - Memory: ConversationBufferMemory, ConversationSummaryMemory - Retrievers: Vector store retrievers, web search retrievers - Output Parsers: Structured output formatting and validation - Callbacks: Logging, monitoring, and debugging capabilities Chain Types: - Sequential Chains: Linear execution flow - Router Chains: Conditional execution based on input - Transform Chains: Data transformation and preprocessing - Map-Reduce Chains: Parallel processing and aggregation Integration Ecosystem: - Vector Databases: Pinecone, Weaviate, ChromaDB, FAISS - LLM Providers: OpenAI, Anthropic, Cohere, Hugging Face - External Tools: Search engines, APIs, databases - Deployment: Cloud platforms and containerization support

LangGraph Multi-Agent Systems

LangGraph Skip to content Our Building Ambient Agents with LangGraph course is now available on LangChain Academy! LangGraph Trusted by companies shaping the future of agents - including Klarna, Replit, Elastic, and more - LangGraph is a low-level orchestration framework for building, managing, and deploying long-running, stateful agents. Get started Install LangGraph: pip install -U langgraph Then, create an agent using prebuilt components: API Reference: create_react_agent # pip install -qU "langchain[anthropic]" to call the model from langgraph.prebuilt import create_react_agent def get_weather(city: str) -> str: """Get weather for a given city.""" return f"It's always sunny in {city}!" agent = create_react_agent(model="anthropic:claude-3-7-sonnet-latest", tools=[get_weather], prompt="You are a helpful assistant") # Run the agent agent.invoke({"messages": [{"role": "user", "content": "what is the weather in sf"}]}) For more information, see the Quickstart. Or, to learn how to build an agent workflow with a customizable architecture, long-term memory, and other complex task handling, see the LangGraph basics tutorials. Core benefits LangGraph provides low-level supporting infrastructure for any long-running, stateful workflow or agent. LangGraph does not abstract prompts or architecture, and provides the following central benefits: Durable execution: Build agents that persist through failures and can run for extended periods, automatically resuming from exactly where they left off. Human-in-the-loop: Seamlessly incorporate human oversight by inspecting and modifying agent state at any point during execution. Comprehensive memory: Create truly stateful agents with both short-term working memory for ongoing reasoning and long-term persistent memory across sessions. Debugging with LangSmith: Gain deep visibility into complex agent behavior with visualization tools that trace execution

paths, capture state transitions, and provide detailed runtime metrics. Production-ready deployment: Deploy sophisticated agent systems confidently with scalable infrastructure designed to handle the unique challenges of stateful, long-running workflows. LangGraph's ecosystem While LangGraph can be used standalone, it also integrates seamlessly with any LangChain product, giving developers a full suite of tools for building agents. To improve your LLM application development, pair LangGraph with: LangSmith - Helpful for agent evals and observability. Debug poor-performing LLM app runs, evaluate agent trajectories, gain visibility in production, and improve performance over time. LangGraph Platform - Deploy and scale agents effortlessly with a purpose-built deployment platform for long running, stateful workflows. Discover, reuse, configure, and share agents across teams - and iterate quickly with visual prototyping in LangGraph Studio. LangChain - Provides integrations and composable components to streamline LLM application development. Note Looking for the JS version of LangGraph? See the JS repo and the JS docs. Additional resources Guides: Quick, actionable code snippets for topics such as streaming, adding memory & persistence, and design patterns (e.g. branching, subgraphs, etc.). Reference: Detailed reference on core classes, methods, how to use the graph and checkpointing APIs, and higher-level prebuilt components. Examples: Guided examples on getting started with LangGraph. LangChain Forum: Connect with the community and share all of your technical questions, ideas, and feedback. LangChain Academy: Learn the basics of LangGraph in our free, structured course. Templates: Pre-built reference apps for common agentic workflows (e.g. ReAct agent, memory, retrieval etc.) that can be cloned and adapted. Case studies: Hear how industry leaders use LangGraph to ship AI applications at scale. Acknowledgements LangGraph is inspired by Pregel and Apache Beam. The public interface draws inspiration from NetworkX. LangGraph is built by LangChain Inc, the creators of LangChain, but can be used without LangChain. Back to top

LangSmith Platform

What is LangSmith? | IBM What is LangSmith? Artificial Intelligence DevOps Publication Date 12 June 2025 How does LangSmith work? LangSmith operates by embedding itself into the LLM application stack to provide visibility, traceability and control at every stage of development and production. Link copied Author Jobit Varughese Technical Content Writer IBM What is LangSmith? One of the biggest challenges in building reliable large language model (LLM) applications is understanding why an artificial intelligence (AI) system fails or behaves unexpectedly, once deployed. Developers often struggle to trace bugs, fine-tune prompts, evaluate performance across edge cases or debug tool use and memory issues in complex agent workflows. LangSmith, developed by the team behind LangChain, offers a robust solution for addressing these challenges. It serves as a dedicated platform for monitoring, debugging and evaluating applications built with large language models. It lets developers inspect traces, monitor performance, test different prompt versions and track how external tools and memory are used in real-time, all within a unified interface designed to make LLM apps more robust and production ready. Understanding LangSmith and LangChain LangChain and LangSmith are tools to support LLM development, but the purpose of each tool varies. LangChain is an open source Python

framework that simplifies the building and deployment of LLM applications. It connects multiple LLM components into structured workflows by using modular building blocks such as chains, agents and memory. These components enable the integration of LLMs with external tools, application programming interfaces (APIs) and data sources to build complex applications. Instead of relying on a single model, it supports chaining together models for tasks such as text understanding, response generation and reasoning, allowing each step to build on the last. LangChain supports prompt engineering through reusable templates and integrates with LangGraph for visually designing workflows. This ability makes it especially powerful for building conversational agents and AI systems that require context handling and logical progression. Moreover, LangSmith is the operational backbone to LangChain's development capabilities. While LangChain helps you build workflows, LangSmith helps ensure that they run smoothly by offering tools for debugging, monitoring and managing complex AI systems. LangSmith provides deep visibility into model behavior, making it easier to identify performance issues, trace errors and optimize responses in real time. It also supports orchestration across multiple models and pipelines, allowing seamless deployment and coordination. LangSmith offers seamless integration with external tools such as TensorFlow, Kubernetes. It can also be integrated with major cloud providers like AWS, GCP and Azure, while also providing robust support for hybrid setups and on-premises deployments. LangSmith supports real-world AI application development, including chatbots and other interactive systems such as AI agents, virtual assistants and conversational interfaces. This capability helps developers streamline their workflows. Together, LangChain and LangSmith simplify the entire development process from prototyping to production. How does LangSmith work? LangSmith operates by embedding itself into the LLM application stack, whether you're using LangChain or building custom pipelines to provide visibility, traceability and control at every stage of development and production. It captures granular data from each LLM interaction and visualizes it, helping developers pinpoint problems, test solutions and optimize performance. The major functions of LangSmith are: Debugging Testing Evaluating Monitoring Debugging LLM applications often involve complex reasoning paths, dynamic tool usage and multistep chains. When errors occur, such as infinite loops, incorrect outputs or tool invocation failures, traditional debugging methods fall short. LangSmith offers detailed, sequential visibility into each interaction with LLMs, helping ensure clear traceability throughout the process. Trace, track and display the step-by-step flow of data through the application by using LangChain Expression Language (LCEL). This visibility helps troubleshoot long response times, errors or unexpected behavior. LangSmith provides rich visualization tools to display LLM call traces, helping developers understand and debug complex workflows easily. Developers can inspect individual prompts and responses, intermediate steps within chains and agents, and tool calls and their corresponding outputs. This fine-grained visibility enables rapid identification and resolution of issues, significantly reducing development time and improving application stability. Testing LLM applications require frequent updates, whether optimizing prompts, adjusting chain logic or changing model parameters. Helping ensure these changes do not introduce regressions is essential. LangSmith supports dataset-driven testing, allowing developers to run predefined or custom test suites

across application versions, compare outputs visually and semantically and identify changes in behavior before deploying to production. This testing facilitates rigorous quality assurance and promotes safe, iterative development. LangSmith's support for automated evaluations enables teams to quickly iterate on prompt designs and model parameters to ensure consistent quality. Evaluating Beyond functional correctness, the quality of LLM-generated outputs must be continuously evaluated against business and user expectations. LangSmith offers both built-in and customizable evaluators to assess performance across various dimensions such as accuracy, relevance and coherence. With LangSmith's evaluation capabilities, teams can benchmark performance across datasets and prompt variations, surface edge cases that degrade user experience and track improvements or regressions with clear metrics. This structured evaluation process helps ensure that LLM systems remain effective, accurate and aligned with intended outcomes. Monitoring Deploying LLM applications into production requires robust monitoring to help ensure consistent performance and immediate incident response. LangSmith delivers end-to-end observability for LLM workflows such as real-time logging of executions, latency and error rates, integration with alerting systems for prompt incident reporting and dashboards that provide insights into usage patterns and system health. This operational intelligence allows engineering teams to proactively manage application behavior, helping ensure reliability and responsiveness in live environments. Real-world deployment monitoring with LangSmith helps teams streamline incident response and maintain robust system health. LangSmith works through a simple Python SDK that helps developers build and manage AI applications easily. It connects with AI models like OpenAI's GPT and uses techniques such as retrieval-augmented generation (RAG) to improve how these models work. By using an API key, developers can track and debug AI agents, including those based on ChatGPT, making sure everything runs smoothly and performs well in generative AI projects. For example, this research presents a LangSmith editor that assists non-native researchers in writing academic papers in English, particularly in the NLP domain. The system offers three main features: text revision suggestions based on rough drafts, text completion conditioned on context and grammatical or spelling error correction.[1] Results demonstrated that LangSmith improves the quality of draft revisions, especially when human and machine collaboration is involved, enabling non-native writers to produce more fluent and stylistically appropriate academic texts. The system enhances diversity and inclusion by lowering language barriers in scientific communication. This example highlights a real-world use case where LangSmith facilitates data science research by improving collaboration between humans and AI in academic writing. Such use cases demonstrate LangSmith's ability to enhance inclusivity and productivity in various AI-driven fields. Factory, a company building AI agents to automate the Software Development Lifecycle (SDLC), uses LangSmith to help ensure secure, reliable LLM operations in enterprise environments.[2] They integrated LangSmith with AWS CloudWatch and gained full traceability across its LLM pipelines, enabling faster debugging and better context management. Using LangSmith's Feedback API, they automated prompt evaluation and refinement based on real user input. This helped double iteration speed and reduced open-to-merge time by 20%, making LangSmith a critical part of their AI development and observability workflow. Benefits and challenges of LangSmith Benefits

All-in-one platform: LangSmith consolidates all core functions-debugging, testing, deployment, monitoring-into a single cohesive platform. Real-world deployment monitoring with LangSmith helps teams streamline incident response and maintain robust system health. Its clean, developer-friendly interface makes it easy to navigate complex workflows and manage projects efficiently without switching between multiple tools. Robust debugging and evaluation: Provides detailed trace analysis, prompt testing and dataset management tools that help pinpoint issues, measure performance and refine LLM behavior with precision. Enterprise-ready scalability: Designed to support high-volume, production-grade applications, making it a strong fit for enterprise teams building and maintaining complex AI systems. Challenges Steep learning curve for beginners: LangSmith can be challenging for beginners, as it demands a solid understanding of LLM tools and DevOps processes, which can limit its accessibility for newcomers. Heavy dependence on LangChain ecosystem: LangSmith is deeply tied to LangChain. While this is great for users of that framework, it might not be as helpful for those using other orchestration tools or custom stacks. Scalability and cost for large-scale projects: For enterprise use, costs can grow with scale, especially when dealing with frequent evaluations, large trace storage or advanced analytics. The choice between LangChain, LangSmith or a combination of both depends on the specific requirements of your LLM application. LangChain is well suited for designing and prototyping complex language model workflows, enabling seamless integration with external tools and APIs. Use LangSmith when you're ready to move into production and need robust tools for debugging, testing, monitoring and maintaining LLM applications at scale. When used together, these platforms provide a comprehensive and scalable solution for building, deploying and maintaining high-quality LLM applications. Footnotes 1 Ito, T., Kuribayashi, T., Hidaka, M., Suzuki, J., & Inui, K. (2020). Langsmith: An interactive academic text revision system. arXiv preprint arXiv:2010.04332. 2 LangChain. (2024, June 19). How Factory used LangSmith to automate their feedback loop and improve iteration speed by 2x. LangChain Blog. https://blog.langchain.dev/customers-factory/ Ebook How to choose the right foundation model Learn how to choose the right approach in preparing datasets and employing foundation models. Read the ebook Related solutions RAG on watsonx.ai Streamline RAG application building. Build, optimize and deploy RAG pipelines with your enterprise knowledge base. Explore RAG on watsonx.ai Artificial intelligence solutions Put AI to work in your business with IBM's industry-leading AI expertise and portfolio of solutions at your side. Explore AI solutions AI consulting and services Reinvent critical workflows and operations by adding AI to maximize experiences, real-time decision-making and business value. Explore AI services Resources Training Take your gen AI skills to the next level Learn fundamental concepts and build your skills with hands-on labs, courses, guided projects, trials and more. Learn generative AI Guide Put AI to work: Driving ROI with gen AI Want to get a better return on your AI investments? Learn how scaling gen AI in key areas drives change by helping your best minds build and deliver innovative new solutions. Read the guide Report From AI projects to profits: How agentic AI can sustain financial returns Learn how organizations are shifting from launching AI in disparate pilots to using it to drive transformation at the core. Read the report Guide The CEO's guide to generative AI Learn how CEOs can balance the value generative AI can create

against the investment it demands and the risks it introduces. Read the guide Training watsonx Developer Hub Support your next project with some of our most commonly used capabilities. Get started and learn more about the supported models that IBM provides. Get started Report The truth about successful generative AI Uncover the benefits of AI platforms that enable foundation model customization through technology, processes, and best practices, to help you easily operationalize the genAI lifecycle. Read the report Report IBM is named a Leader in Data Science & Machine Learning Learn why IBM has been recognized as a Leader in the 2025 Gartner Magic QuadrantTM for Data Science and Machine Learning Platforms. Read the report Report AI in Action 2024 We surveyed 2,000 organizations about their AI initiatives to discover what's working, what's not and how you can get ahead. Read the report AI models Explore IBM Granite IBM GraniteTM is our family of open, performant and trusted AI models tailored for business and optimized to scale your AI applications. Explore language, code, time series and guardrail options. Meet Granite Ebook How to choose the right foundation model Learn how to select the most suitable AI foundation model for your use case. Read the ebook Guide How to thrive in this new era of AI with trust and confidence Dive into the 3 critical elements of a strong AI strategy: creating a competitive edge, scaling AI across the business and advancing trustworthy AI. Read the guide Take the next step Get one-stop access to capabilities that span the AI development lifecycle. Produce powerful AI solutions with user-friendly interfaces, workflows and access to industry-standard APIs and SDKs. Explore watsonx.ai Book a live demo

RAG Implementation

Retrieval-augmented generation - Wikipedia Jump to content From Wikipedia, the free encyclopedia Type of information retrieval using LLMs Retrieval-augmented generation (RAG) is a technique that enables large language models (LLMs) to retrieve and incorporate new information.[1] With RAG, LLMs do not respond to user queries until they refer to a specified set of documents. These documents supplement information from the LLM's pre-existing training data.[2] This allows LLMs to use domain-specific and/or updated information that is not available in the training data.[2][3] For example, this helps LLM-based chatbots access internal company data or generate responses based on authoritative sources. RAG improves large language models (LLMs) by incorporating information retrieval before generating responses.[4] Unlike traditional LLMs that rely on static training data, RAG pulls relevant text from databases, uploaded documents, or web sources.[1] According to Ars Technica, "RAG is a way of improving LLM performance, in essence by blending the LLM process with a web search or other document look-up process to help LLMs stick to the facts." This method helps reduce AI hallucinations,[4][5] which have caused chatbots to describe policies that don't exist, or recommend nonexistent legal cases to lawyers that are looking for citations to support their arguments.[6] RAG also reduces the need to retrain LLMs with new data, saving on computational and financial costs.[1][7] Beyond efficiency gains, RAG also allows LLMs to include sources in their responses, so users can verify the cited sources. This provides greater transparency, as users can cross-check retrieved content to ensure accuracy and

relevance. The term RAG was first introduced in a 2020 research paper[4] from Meta.[8][3] RAG and LLM Limitations[edit] LLMs can provide incorrect information. For example, when Google first demonstrated its LLM tool "Google Bard", the LLM provided incorrect information about the James Webb Space Telescope. This error contributed to a $100 billion decline in the company's stock value.[6] RAG is used to prevent these errors, but it does not solve all the problems. For example, LLMs can generate misinformation even when pulling from factually correct sources if they misinterpret the context.[9] MIT Technology Review gives the example of an AI-generated response stating, "The United States has had one Muslim president, Barack Hussein Obama." The model retrieved this from an academic book rhetorically titled Barack Hussein Obama: America's First Muslim President? The LLM did not "know" or "understand" the context of the title, generating a false statement.[2] LLMs with RAG are programmed to prioritize new information. This technique has been called "prompt stuffing." Without prompt stuffing, the LLM's input is generated by a user; with prompt stuffing, additional relevant context is added to this input to guide the model's response. This approach provides the LLM with key information early in the prompt, encouraging it to prioritize the supplied data over pre-existing training knowledge.[10] Process[edit] Retrieval-augmented generation (RAG) enhances large language models (LLMs) by incorporating an information-retrieval mechanism that allows models to access and utilize additional data beyond their original training set. AWS states, "RAG allows LLMs to retrieve relevant information from external data sources to generate more accurate and contextually relevant responses" ("indexing").[11] This approach reduces reliance on static datasets, which can quickly become outdated. When a user submits a query, RAG uses a document retriever to search for relevant content from available sources before incorporating the retrieved information into the model's response ("retrieval").[12] Ars Technica notes that "when new information becomes available, rather than having to retrain the model, all that's needed is to augment the model's external knowledge base with the updated information" ("augmentation").[6] By dynamically integrating relevant data, RAG enables LLMs to generate more informed and contextually grounded responses ("generation").[5] IBM states that "in the generative phase, the LLM draws from the augmented prompt and its internal representation of its training data to synthesize an engaging answer tailored to the user in that instant.[1] RAG key stages[edit] Indexing[edit] Typically, the data to be referenced is converted into LLM embeddings, numerical representations in the form of a large vector space.[9] RAG can be used on unstructured (usually text), semi-structured, or structured data (for example knowledge graphs).[13] These embeddings are then stored in a vector database to allow for document retrieval.[14] Overview of RAG process, combining external documents and user input into an LLM prompt to get tailored output Retrieval[edit] Given a user query, a document retriever is first called to select the most relevant documents that will be used to augment the query.[2][4] This comparison can be done using a variety of methods, which depend in part on the type of indexing used.[1][13] Augmentation[edit] The model feeds this relevant retrieved information into the LLM via prompt engineering of the user's original query.[11][15] Newer implementations (as of 2023[update]) can also incorporate specific augmentation modules with abilities such as expanding queries into multiple domains and using memory and self-improvement to learn from previous

retrievals.[13] Generation[edit] Finally, the LLM can generate output based on both the query and the retrieved documents.[2][16] Some models incorporate extra steps to improve output, such as the re-ranking of retrieved information, context selection, and fine-tuning.[13] Improvements[edit] This section possibly contains original research. This section includes uncited claims and arXiv preprints, making it original research under WP:OR and WP:V without reliable sources. Please improve it by verifying the claims made and adding inline citations. Statements consisting only of original research should be removed. (March 2025) (Learn how and when to remove this message) Improvements to the basic process above can be applied at different stages in the RAG flow. Encoder[edit] These methods focus on the encoding of text as either dense or sparse vectors. Sparse vectors, which encode the identity of a word, are typically dictionary-length and contain mostly zeros. Dense vectors, which encode meaning, are more compact and contain fewer zeros. Various enhancements can improve the way similarities are calculated in the vector stores (databases).[17] Performance improves by optimizing how vector similarities are calculated. Dot products enhance similarity scoring, while approximate nearest neighbor (ANN) searches improve retrieval efficiency over K-nearest neighbors (KNN) searches.[18] Accuracy may be improved with Late Interactions, which allow the system to compare words more precisely after retrieval. This helps refine document ranking and improve search relevance.[19] Hybrid vector approaches may be used to combine dense vector representations with sparse one-hot vectors, taking advantage of the computational efficiency of sparse dot products over dense vector operations.[17] Other retrieval techniques focus on improving accuracy by refining how documents are selected. Some retrieval methods combine sparse representations, such as SPLADE, with query expansion strategies to improve search accuracy and recall.[20] Retriever-centric methods[edit] These methods aim to enhance the quality of document retrieval in vector databases: Pre-training the retriever using the Inverse Cloze Task (ICT), a technique that helps the model learn retrieval patterns by predicting masked text within documents.[21] Progressive data augmentation, as used in Diverse Augmentation for Generalizable Dense Retrieval (DRAGON), improves dense retrieval by sampling difficult negative examples during training.[22] Supervised retriever optimization aligns retrieval probabilities with the generator model's likelihood distribution. This involves retrieving the top-k vectors for a given prompt, scoring the generated response's perplexity, and minimizing KL divergence between the retriever's selections and the model's likelihoods to refine retrieval.[23] Reranking techniques can refine retriever performance by prioritizing the most relevant retrieved documents during training.[24][12] Language model[edit] Retro language model for RAG. Each Retro block consists of Attention, Chunked Cross Attention, and Feed Forward layers. Black-lettered boxes show data being changed, and blue lettering shows the algorithm performing the changes. By redesigning the language model with the retriever in mind, a 25-time smaller network can get comparable perplexity as its much larger counterparts.[25] Because it is trained from scratch, this method (Retro) incurs the high cost of training runs that the original RAG scheme avoided. The hypothesis is that by giving domain knowledge during training, Retro needs less focus on the domain and can devote its smaller weight resources only to language semantics. The redesigned language model is shown here. It has been reported that Retro is not reproducible, so

modifications were made to make it so. The more reproducible version is called Retro++ and includes in-context RAG.[26] Chunking[edit] This section does not cite any sources. Please help improve this section by adding citations to reliable sources. Unsourced material may be challenged and removed.Find sources: "Retrieval-augmented generation" - news newspapers books scholar JSTOR (October 2024) (Learn how and when to remove this message) Chunking involves various strategies for breaking up the data into vectors so the retriever can find details in it.[14] Different data styles have patterns that correct chunking can take advantage of. Three types of chunking strategies are: Fixed length with overlap. This is fast and easy. Overlapping consecutive chunks helps to maintain semantic context across chunks. Syntax-based chunks can break the document up into sentences. Libraries such as spaCy or NLTK can also help. File format-based chunking. Certain file types have natural chunks built in, and it's best to respect them. For example, code files are best chunked and vectorized as whole functions or classes. HTML files should leave <table> or base64 encoded <img> elements intact. Similar considerations should be taken for pdf files. Libraries such as Unstructured or Langchain can assist with this method. Knowledge graphs[edit] Rather than using documents as a source to vectorize and retrieve from, Knowledge Graphs can be used. One can start with a set of documents, books, or other bodies of text, and convert them to a knowledge graph using one of many methods, including language models. Once the knowledge graph is created, subgraphs can be vectorized, stored in a vector database, and used for retrieval as in plain RAG. The advantage here is that graphs has more recognizable structure than strings of text and this structure can help retrieve more relevant facts for generation. Sometimes this approach is called GraphRAG.[citation needed] Hybrid search[edit] Sometimes vector database searches can miss key facts needed to answer a user's question. One way to mitigate this is to do a traditional text search, add those results to the text chunks linked to the retrieved vectors from the vector search, and feed the combined hybrid text into the language model for generation.[citation needed] Late-interaction Search[edit] Since vector search relies on embedding individual chunks, a lot of granular, token-level information cannot be obtained via pure hybrid of vector search. For higher accuracy, one can create embeddings out of individual tokens instead and compute the Chamfer distance between them. This leads to significantly better results at the cost of speed. Modern solutions such as Morphik make this technique scalable by using a combination of software and hardware acceleration. Evaluation and Benchmarks[edit] RAG systems are commonly evaluated using benchmarks designed to test both retrieval accuracy and generative quality. Popular datasets include BEIR, a suite of information retrieval tasks across diverse domains, and Natural Questions or Google QA for open-domain QA. In high-stakes domains like law and healthcare, domain-specific benchmarks are increasingly used. For instance, LegalBench-RAG[27] is an open-source benchmark designed to test retrieval quality over legal documents. It evaluates recall and precision for different RAG pipelines using real-world legal questions and documents. Challenges[edit] RAG is not a complete solution to the problem of hallucinations in LLMs. According to Ars Technica, "It is not a direct solution because the LLM can still hallucinate around the source material in its response."[6] While RAG improves the accuracy of large language models (LLMs), it does not eliminate all challenges. One limitation is that while RAG reduces

the need for frequent model retraining, it does not remove it entirely. Additionally, LLMs may struggle to recognize when they lack sufficient information to provide a reliable response. Without specific training, models may generate answers even when they should indicate uncertainty. According to IBM, this issue can arise when the model lacks the ability to assess its own knowledge limitations.[1] RAG systems may retrieve factually correct but misleading sources, leading to errors in interpretation. In some cases, an LLM may extract statements from a source without considering its context, resulting in an incorrect conclusion.[12] Additionally, when faced with conflicting information RAG models may struggle to determine which source is accurate. The worst case outcome of this limitation is that the model may combine details from multiple sources producing responses that merge outdated and updated information in a misleading manner. According to the MIT Technology Review, these issues occur because RAG systems may misinterpret the data they retrieve.[2] References[edit] ^ a b c d e f "What is retrieval-augmented generation?". IBM. 22 August 2023. Retrieved 7 March 2025. ^ a b c d e f "Why Google's AI Overviews gets things wrong". MIT Technology Review. 31 May 2024. Retrieved 7 March 2025. ^ a b Singhal, Rahul (Nov 30, 2023). "The Power Of RAG: How Retrieval-Augmented Generation Enhances Generative AI". Forbes. ^ a b c d Kiela Douwe, Lewis Patrick, Perez Ethan, Piktus Aleksandra, Petroni Fabio, Karpukhin Vladimir, Goyal Naman, Kuttler Heinrich, Lewis Mike, Yih Wen-Tau, Rocktaschel Tim, Riedel Sebastian (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. pp. 9459-9474. arXiv:2005.11401. ISBN 978-1-7138-2954-6.{{cite book}}: CS1 maint: multiple names: authors list (link) ^ a b Turow Jon, Kiela Douwe (March 26, 2025). "RAG Inventor Talks Agents, Grounded AI, and Enterprise Impact". Madrona. ^ a b c d "Can a technology called RAG keep AI models from making stuff up?". Ars Technica. 6 June 2024. Retrieved 7 March 2025. ^ Mishi, Javed. "Retrieval-Augmented Generation for Enterprise Search Systems". Nextbridge. Hajra Naeem. Retrieved 11 July 2025. ^ "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks". ai.meta.com. 2020. ^ a b Xu, Sherlock (January 25, 2024). "Understanding Retrieval-Augmented Generation: Part 1". www.bentoml.com. ^ "Mitigating LLM hallucinations in text summarisation". BBC. 20 June 2024. Retrieved 7 March 2025. ^ a b "What is RAG? - Retrieval-Augmented Generation AI Explained - AWS". Amazon Web Services, Inc. Retrieved 16 July 2024. ^ a b c Kiela Douwe, Turck Matt (March 6, 2025). "Top AI Researcher on GPT 4.5, DeepSeek and Agentic RAG | Douwe Kiela, CEO, Contextual AI". YouTube. ^ a b c d Gao, Yunfan; Xiong, Yun; Gao, Xinyu; Jia, Kangxiang; Pan, Jinliu; Bi, Yuxi; Dai, Yi; Sun, Jiawei; Wang, Meng; Wang, Haofen (2023). "Retrieval-Augmented Generation for Large Language Models: A Survey". arXiv:2312.10997 [cs.CL]. ^ a b Sankar, Shrinivasan (Feb 13, 2024). "Retrieval Augmented Generation(RAG) - A quick and comprehensive introduction". ai-bites.net. ^ Kiela Douwe, Ho Alan (Oct 13, 2023). "Where did Retrieval Augmented Generation come from, and where is it going?". YouTube. ^ Lewis, Patrick; Perez, Ethan; Piktus, Aleksandra; Petroni, Fabio; Karpukhin, Vladimir; Goyal, Naman; Kuttler, Heinrich; Lewis, Mike; Yih, Wen-tau; Rocktaschel, Tim; Riedel, Sebastian; Kiela, Douwe (2020). "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks". Advances in Neural Information Processing Systems. 33. Curran Associates, Inc.: 9459-9474. arXiv:2005.11401. ^ a b Luan, Yi; Eisenstein, Jacob; Toutanova, Kristina; Collins, Michael (26 April 2021). "Sparse, Dense, and Attentional Representations for

Text Retrieval". Transactions of the Association for Computational Linguistics. 9: 329-345. arXiv:2005.00181. doi:10.1162/tacl_a_00369. Retrieved 15 March 2025. ^ "Information retrieval". Microsoft. 10 January 2025. Retrieved 15 March 2025. ^ Khattab, Omar; Zaharia, Matei (2020). ""ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT"". doi:10.1145/3397271.3401075. ^ Wang, Yup; Conroy, John M.; Molino, Neil; Yang, Julia; Green, Mike (2024). "Laboratory for Analytic Sciences in TREC 2024 Retrieval Augmented Generation Track". NIST TREC 2024. Retrieved 15 March 2025. ^ Lee, Kenton; Chang, Ming-Wei; Toutanova, Kristina (2019). ""Latent Retrieval for Weakly Supervised Open Domain Question Answering"" (PDF). ^ Lin, Sheng-Chieh; Asai, Akari (2023). ""How to Train Your DRAGON: Diverse Augmentation Towards Generalizable Dense Retrieval"" (PDF). ^ Shi, Weijia; Min, Sewon; Yasunaga, Michihiro; Seo, Minjoon; James, Rich; Lewis, Mike; Zettlemoyer, Luke; Yih, Wen-tau (June 2024). "REPLUG: Retrieval-Augmented Black-Box Language Models. In Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 8371-8384, Mexico City, Mexico. Association for Computational Linguistics". ACL Anthology (Publisher: Association for Computational Linguistics): 8371-8384. arXiv:2301.12652. doi:10.18653/v1/2024.naacl-long.463. Retrieved 16 March 2025. ^ Ram, Ori; Levine, Yoav; Dalmedigos, Itay; Muhlgay, Dor; Shashua, Amnon; Leyton-Brown, Kevin; Shoham, Yoav (2023). "In-Context Retrieval-Augmented Language Models. Transactions of the Association for Computational Linguistics, 11:1316-1331". ACL Anthology (Publisher: MIT Press). arXiv:2302.00083. doi:10.1162/tacl_a_00605. Retrieved 16 March 2025. ^ Borgeaud, Sebastian; Mensch, Arthur (2021). ""Improving language models by retrieving from trillions of tokens"" (PDF). ^ Wang, Boxin; Ping, Wei (2023). ""Shall We Pretrain Autoregressive Language Models with Retrieval? A Comprehensive Study"" (PDF). ^ LegalBench-RAG (2024) vteGenerative AIConcepts Autoencoder Deep learning Fine-tuning Foundation model Generative adversarial network Generative pre-trained transformer Large language model Model Context Protocol Neural network Prompt engineering Reinforcement learning from human feedback Retrieval-augmented generation Self-supervised learning Stochastic parrot Synthetic data Top-p sampling Transformer Variational autoencoder Vibe coding Vision transformer Waluigi effect Word embedding ModelsText Character.ai ChatGLM Claude DeepSeek Ernie Gemini GPT 1 2 3 J ChatGPT 4 4o o1 o3 4.5 4.1 o4-mini Grok Hunyuan Turbo S Kimi Llama Microsoft Copilot MiniMax-M1 Magistral Medium Qwen Yi-Lightning Coding Claude Code Cursor Devstral GitHub Copilot Granite Code Kimi-Dev Replit Xcode Image Aurora Firefly Flux GPT Image 1 Ideogram Imagen Leonardo Midjourney Recraft Seedream Stable Diffusion Video Dream Machine Hailuo AI Kling Midjourney Video Runway Gen Seedance Sora Veo Wan Speech 15.ai ElevenLabs Speech-02 WaveNet Music Endel Lyria Riffusion Suno AI Udio Agents AutoGLM AutoGPT Devin AI Manus OpenAI Codex Operator Replit Agent Companies 01.AI Aleph Alpha Anthropic Baichuan Canva Cognition AI Cohere Contextual AI DeepSeek Google DeepMind HeyGen Hugging Face Krikey AI Kuaishou Luma Labs Meta AI MiniMax Mistral AI Moonshot AI OpenAI Runway Safe Superintelligence Scale AI Stability AI Synthesia Thinking Machines Lab xAI Zhipu AI Category Commons vteArtificial intelligence (AI)History (timeline)Concepts Parameter Hyperparameter Loss functions Regression Bias-variance tradeoff Double descent Overfitting Clustering Gradient descent SGD

Quasi-Newton method Conjugate gradient method Backpropagation Attention Convolution Normalization Batchnorm Activation Softmax Sigmoid Rectifier Gating Weight initialization Regularization Datasets Augmentation Prompt engineering Reinforcement learning Q-learning SARSA Imitation Policy gradient Diffusion Latent diffusion model Autoregression Adversary RAG Uncanny valley RLHF Self-supervised learning Reflection Recursive self-improvement Hallucination Word embedding Vibe coding Applications Machine learning In-context learning Artificial neural network Deep learning Language model Large language model NMT Reasoning language model Model Context Protocol Intelligent agent Artificial human companion Humanity's Last Exam Artificial general intelligence (AGI) ImplementationsAudio-visual AlexNet WaveNet Human image synthesis HWR OCR Computer vision Speech synthesis 15.ai ElevenLabs Speech recognition Whisper Facial recognition AlphaFold Text-to-image models Aurora DALL-E Firefly Flux Ideogram Imagen Midjourney Recraft Stable Diffusion Text-to-video models Dream Machine Runway Gen Hailuo AI Kling Sora Veo Music generation Suno AI Udio Text Word2vec Seq2seq GloVe BERT T5 Llama Chinchilla AI PaLM GPT 1 2 3 J ChatGPT 4 4o o1 o3 4.5 4.1 o4-mini Claude Gemini chatbot Grok LaMDA BLOOM DBRX Project Debater IBM Watson IBM Watsonx Granite PanGu- DeepSeek Qwen Decisional AlphaGo AlphaZero OpenAI Five Self-driving car MuZero Action selection AutoGPT Robot control People Alan Turing Warren Sturgis McCulloch Walter Pitts John von Neumann Claude Shannon Shun'ichi Amari Kunihiko Fukushima Takeo Kanade Marvin Minsky John McCarthy Nathaniel Rochester Allen Newell Cliff Shaw Herbert A. Simon Oliver Selfridge Frank Rosenblatt Bernard Widrow Joseph Weizenbaum Seymour Papert Seppo Linnainmaa Paul Werbos Geoffrey Hinton John Hopfield Jurgen Schmidhuber Yann LeCun Yoshua Bengio Lotfi A. Zadeh Stephen Grossberg Alex Graves James Goodnight Andrew Ng Fei-Fei Li Ilya Sutskever Alex Krizhevsky Ian Goodfellow Demis Hassabis David Silver Andrej Karpathy Ashish Vaswani Noam Shazeer Aidan Gomez Francois Chollet Architectures Neural Turing machine Differentiable neural computer Transformer Vision transformer (ViT) Recurrent neural network (RNN) Long short-term memory (LSTM) Gated recurrent unit (GRU) Echo state network Multilayer perceptron (MLP) Convolutional neural network (CNN) Residual neural network (RNN) Highway network Mamba Autoencoder Variational autoencoder (VAE) Generative adversarial network (GAN) Graph neural network (GNN) Portals Technology Category Artificial neural networks Machine learning List Companies Projects Retrieved from "https://en.wikipedia.org/w/index.php?title=Retrieval-augmented_generation&oldid=1300820373" Categories: Large language modelsNatural language processingInformation retrieval systemsGenerative artificial intelligenceHidden categories: CS1 maint: multiple names: authors listArticles with short descriptionShort description is different from WikidataArticles containing potentially dated statements from 2023All articles containing potentially dated statementsArticles that may contain original research from March 2025All articles that may contain original researchArticles needing additional references from October 2024All articles needing additional referencesAll articles with unsourced statementsArticles with unsourced statements from April 2025Articles with unsourced statements from February 2025 Search Search Retrieval-augmented generation 14 languages Add topic

Model Context Protocol

#14: What Is MCP, and Why Is Everyone - Suddenly!- Talking About It? Back to Articles
#14: What Is MCP, and Why Is Everyone - Suddenly!- Talking About It? Community Article
Published March 17, 2025 Upvote 313 +307 Ksenia Se Kseniase Follow everything you need to know about Model Context Protocol "Even the most sophisticated models are constrained by their isolation from data - trapped behind information silos and legacy systems."
Anthropic, on why context integration matters Large language models (LLMs) today are incredibly smart in a vacuum, but they struggle once they need information beyond what's in their frozen training data. For AI agents to be truly useful, they must access the right context at the right time - whether that's your files, knowledge bases, or tools - and even take actions like updating a document or sending an email based on that context. Historically, connecting an AI model to all these external sources has been a messy, ad-hoc affair. Developers had to write custom code or use specialized plugins for each data source or API. This made "wire together" integrations brittle and hard to scale. To simplify that, Anthropic came up with Model Context Protocol (MCP) - an open standard designed to bridge AI assistants with the world of data and tools, to plug in many different sources of context. They announced it in November 2024. The reaction was sort of blah. But now MCP is trending, already passing Langchain and promising to overcome OpenAPI and CrewAI pretty soon. Major AI players and open-source communities are rallying around MCP, seeing it as a potential game-changer for building agentic AI systems. Why? In this article, we'll dive deep into MCP - why it's a hot topic right now, how MCP enables the shift toward more integrated, context-aware AI, its place in agentic workflows, and the under-the-radar details that developers, researchers, AI engineers, and tech executives should know. We'll also explore some innovative applications of MCP that few have attempted. Overall, it's a great starting guide, but also useful for those who have already experimented with MCP and want to learn more. Dive in! Turing Post is on Hugging Face as a resident -> click to follow! UPDATE: if you are interested in protocols, you might also want to read our deep dive into A2A What's in today's episode? Why Is MCP Making Waves Now (and Not Last November)? So, What Is MCP and How Does It Work? Technical Overview of MCP How Do I Actually Get Started with MCP? Before MCP, How Were AI Systems Handling Context And Tool Access? Is MCP a Silver Bullet and Solve-It-All? MCP in Agentic Orchestration and Its Place in the Agentic Workflow New Possibilities Unlocked by MCP Concluding Thoughts Resources to Dive Deeper Why Is MCP Making Waves Now (and Not Last November)? MCP was first open-sourced and announced by Anthropic in late November 2024. At the time, it was an exciting idea but not that many noticed it and took seriously. It's in early 2025 that MCP has really surged into the AI community's consciousness. There are a few big reasons for this recent buzz: Integration Problem Solver: AI agents and agentic workflows became major buzzwords in 2023-2024, but their Achilles' heel remained: integrating these agents with real-world business systems and data. Initially, much attention went to model capabilities and prompt techniques, not integration. MCP squarely addresses this gap by defining "how to connect existing data sources" (file systems, databases, APIs, etc.) into AI workflows. As people digested this, MCP started to be seen as the missing puzzle piece for serious, production-ready AI

agents. (That's one of the takes from HumanX conference: In recent years, we've primarily been focused on building individual AI models, each specialized for specific tasks. But as complexity and demands grow, a shift is happening towards integrated systems - orchestrations of multiple specialized models, software components, APIs, data sources, and interfaces working cohesively.) Community and Adoption: In just a few months, MCP went from concept to a growing ecosystem. Early adopters included companies like Block (Square), Apollo, Zed, Replit, Codeium, and Sourcegraph, who began integrating MCP to enhance their platforms. Fast forward to 2025, and the ecosystem has exploded - by February, there were over 1,000 community-built MCP servers (connectors) available. Clearly, MCP has struck a chord as the industry moves toward more integrated and context-aware AI. This network effect makes MCP even more attractive: the more tools available via MCP, the more useful it is to adopt the standard. De Facto Standard Momentum: Unlike yet another proprietary SDK or one-off framework, MCP is open and model-agnostic, and it's backed by a major AI player. This means any AI model (Claude, GPT-4, open-source LLMs, etc.) can use MCP, and any developer or company can create an MCP integration without permission. Many in the community now see MCP as the likely winner in the race to standardize how AI systems connect to external data (much like how USB, HTTP, or ODBC became ubiquitous standards in their domains). Rapid Evolution and Education: Anthropic didn't just release MCP and walk away; they have been actively improving it and educating developers. During the recent AI Summit, Anthropic's Mahesh Murag delivered a workshop that went viral, accelerating MCP adoption. (Remember, all links for further learning are included at the end of the article.) So, What Is MCP and How Does It Work? MCP lays out clear rules for how AI can find, connect to, and use external tools - whether it's querying a database or running a command. This lets models go beyond their training data, making them more flexible and aware of the world around them. Technical Overview of MCP: One striking feature is MCP's dynamic discovery - AI agents automatically detect available MCP servers and their capabilities, without hard-coded integrations. For example, if you spin up a new MCP server (like a CRM), agents can immediately recognize and use it via a standardized API, offering flexibility traditional approaches can't match. How do I actually get started with MCP? The best place to start is the official MCP documentation and repository. Anthropic open-sourced the spec and provided SDKs (in languages like Python and now even Java). The steps typically are: Run or install an MCP server for the tool or data source you care about. Anthropic has an open-source repo of pre-built servers for popular systems (Google Drive, Slack, Git, databases, etc.). You can install these and configure them (often just running a command with your credentials or keys). Set up the MCP client in your AI app. If you're using Claude's app, you can add the server in the UI. If you're coding your own agent, use the MCP SDK to connect to the server (providing the address/port). Once you've enabled the MCP services in your client, the client will pick on the additional functionality provided: additional tools, resources and prompt templates. Invoke and iterate. The model/agent can now call the MCP tool actions as needed. Make sure to monitor logs to see that it's calling the servers correctly. You'll see requests hitting the MCP server and responses coming back. For a quick start, Anthropic recommends trying the Claude Desktop integration (if you have access) or running the example servers and

using their provided quickstart guide. The community is also very active - there is a rapidly expanding catalog of MCP servers. Some of the popular ones include connectors for Google services (Drive, Gmail, Calendar), Slack (chat and file access), GitHub/Git (for code repositories), databases like Postgres, web browsers or Puppeteer (to browse web pages), and many more. Many servers are listed in community directories (some developers have created sites to index them). The official MCP GitHub also hosts a bunch of connector implementations to get you started. And if you have a niche tool that isn't covered, you can build your own MCP server using the SDK - often it's just a thin wrapper around that tool's API, exposing a function in the MCP format. We thank Will Schenk for clarifying a few things about MCP and how to start with it. He shared this quick hands-on walkthrough with Tezlab's Tesla monitoring service to demonstrate MCP at work. Before MCP, How Were AI Systems Handling Context And Tool Access? Let's briefly look at the traditional approaches to giving AI external knowledge or actions, and how MCP differs: Custom API Integrations (One-off Connectors): The most common method has been writing custom code or using SDKs for each service. For example, if you wanted your AI agent to access Google Drive and a SQL database, you'd integrate Google's API and a database driver separately, each with its own authentication, data format, and quirks. Pain in the neck! MCP, by contrast, gives a single "key" (protocol) that can unlock many doors, and new MCP servers can be added without changing the client. Language Model Plugins (OpenAI Plugins, etc.): Another approach introduced in 2023 was providing the model a standardized plugin specification (often an OpenAPI schema) so it could call external APIs in a controlled way (e.g. the ChatGPT Plugins system). While conceptually similar to MCP (standardizing tool access), these were proprietary and limited - each plugin still needed to be built and hosted individually, and only certain platforms (like ChatGPT or Bing Chat) could use them. Plugins also tended to focus on one-way data retrieval (the model calls an API and gets info) rather than maintaining an ongoing interactive session. MCP distinguishes itself by being open-source and universal (anyone can implement it, not tied to one AI provider) and by supporting rich two-way interactions. It's like a dialogue between the AI and tools, whereas plugins were often stateless question-answer calls. Tool Use via Frameworks (LangChain tools, Agents): Agent orchestration libraries like LangChain popularized the idea of giving models "tools" (functions) with descriptions. For example, you might have a search() tool or a calculate() tool, and the agent (via the LLM) decides when to invoke them. This is powerful, but each tool still required custom implementation under the hood - LangChain's library grew to 500+ tools implemented in a consistent interface, yet developers still had to wire up those tools or ensure they fit their needs. MCP can be seen as complementary here: it provides a standardized interface for the implementation of tools. In fact, you can think of MCP servers as a library of ready-made tools that any agent can use. The difference is where the standardization lies. LangChain created a developer-facing standard (its Tool class interface) to integrate tools into an agent's code. MCP creates a model-facing standard - the running AI agent itself can discover and use any MCP-defined tool at runtime. This means even if you don't custom-build an agent's code for a particular tool, the model can integrate it on the fly. In practice, these ideas are converging: for example, LangChain's team (when noticed the surge of MCP) provided an adapter so that all those

MCP servers (connectors) can be treated as LangChain tools easily. So an agent built inLLangChain or other frameworks can call MCP tools just like any other, benefiting from the growing MCP ecosystem. Retrieval-Augmented Generation (RAG) and Vector Databases: A prevalent way to supply context to LLMs is to use a retriever that searches a knowledge base (documents, embeddings) and injects the top results into the prompt. This addresses the knowledge cutoff or limited memory of models. However, RAG usually deals with static text snippets and doesn't inherently let the model perform actions or queries beyond what's indexed. MCP can actually work alongside RAG - for instance, an MCP server could interface with a vector database or search engine, allowing the model to issue search queries as a tool rather than implicitly relying on retrieval every prompt. One could argue MCP is a more general mechanism: where RAG gives passive context, MCP lets the model actively fetch or act on context through defined channels. In scenarios where up-to-date or interactive data is needed (say, querying a live database or posting an update), MCP extends beyond just retrieving text - it can trigger operations. Is MCP a Silver Bullet and Solve-It-All? Of course, MCP is not a silver bullet, it is an extremely convenient integration layer. But like any emerging technology, it introduces its own set of complexities and challenges that developers and organizations must consider before adopting it at scale: One of the primary concerns is the added overhead of managing multiple tool servers. Running and maintaining connections to these local servers can be cumbersome, particularly in production environments where uptime, security, and scalability are paramount. MCP's initial implementation was designed for local and desktop use, which raises questions about how well it translates to cloud-based architectures and multi-user scenarios. Developers have proposed making MCP more stateless and adaptable to distributed environments, but this remains an ongoing challenge. Another issue lies in tool usability. Just because MCP expands an AI model's toolset does not necessarily mean the model will use those tools effectively. Previous agent-based frameworks have demonstrated that AI models can struggle with tool selection and execution. MCP attempts to mitigate this by providing structured tool descriptions and specifications, but success still hinges on the quality of these descriptions and the AI's ability to interpret them correctly. The community-driven approach, as highlighted by LangChain's founder Harrison Chase, suggests that well-documented tools can enhance usability, but this is still an area of ongoing refinement. Beyond implementation hurdles, MCP's maturity is also a consideration. As a relatively new technology, it is subject to rapid changes and evolving standards. This can lead to breaking changes, requiring frequent updates to servers and clients. While the core concept of MCP appears stable, developers should anticipate and prepare for version upgrades and evolving best practices. Compatibility is another limiting factor. Currently, MCP has first-class support within Anthropic's ecosystem (e.g., Claude), but broader adoption remains uncertain. Other AI providers may not natively support MCP, requiring additional adapters or custom integrations. Until MCP gains wider acceptance across AI platforms, its utility will be somewhat constrained. For simpler applications, MCP may even be overkill. If an AI model only needs to access one or two straightforward APIs, direct API calls might be a more efficient solution than implementing MCP. The learning curve associated with MCP's messaging system and server setup means that its benefits need to be weighed against its

complexity. Security and monitoring also present ongoing challenges. Since MCP acts as an intermediary, it necessitates robust authentication and permission controls to prevent unauthorized access. Open-source initiatives like MCP Guardian have emerged to address these concerns by logging requests and enforcing policies, but securing MCP in enterprise environments remains a work in progress. Overall, none of these limitations are show-stoppers, but it's wise to start with experimental or non-critical deployments to get a feel for it. One of the best things about MCP - the engaged community. Since it's open, issues you face can be discussed and addressed collaboratively. MCP in Agentic Orchestration and Its Place in the Agentic Workflow In previous articles, we explored the building blocks of autonomous agents: Profiling (identity and context), Knowledge, Memory, Reasoning/Planning, Reflection, and Action. An agent needs to observe and understand its environment (profile/knowledge), remember past interactions (memory), plan its moves (reasoning), take actions (execute tool calls or outputs), then reflect and learn. Where does MCP come in? MCP is not itself an "agent framework"; rather, it acts as a standardized integration layer for agents. MCP is all about the Action part - specifically, giving agents a standardized way to perform actions involving external data or tools. It provides the plumbing that connects an AI agent to the outside world in a secure, structured manner. Without MCP (or something like it), every time an agent needs to do something in the world - whether fetching a file, querying a database, or invoking an API - developers would have to wire up a custom integration or use ad-hoc solutions. That's like building a robot but having to custom-craft each finger to grasp different objects - tedious and not scalable. It's important to highlight again that MCP is not an orchestration engine or agent brain by itself. Rather, it's an integration layer within an agentic architecture. It complements agent orchestration tools like LangChain, LangGraph, CrewAI, or LlamaIndex by serving as a unified "toolbox" from which AI agents can invoke external actions. Instead of replacing orchestration - which determines when and why an agent uses a tool - MCP defines how these tools are called and information exchanged. It is akin to a standardized API gateway for agents, reducing integration complexity from an "NM" to an "N+M" problem by allowing universal compatibility between clients (agents) and servers (tools). Ultimately, MCP streamlines the integration of external functionalities, making agents more versatile, adaptable, and capable of performing sophisticated tasks across diverse contexts. New Possibilities Unlocked by MCP MCP is still new, and its full potential is just being explored. The first wave of use cases is obvious - connecting enterprise data to chat assistants or enhancing coding agents with repository access. But some emerging applications could take AI agents to the next level. Multi-Step, Cross-System Workflows Agentic systems often need to coordinate across platforms. Say an AI plans an event: it checks your calendar, books a venue, emails guests, arranges travel, and updates a budget sheet. Right now, this requires stitching APIs together manually. With MCP, all these actions happen through a single interface. The agent calls a series of MCP tools (one for each task), keeping shared context across them-no lost threads, no custom integrations. Agents That Understand Their Environment (including Robotics) Beyond tool access, MCP can enable AI agents embedded in smart environments - whether in a smart home or an operating system. An AI assistant could interact with sensors, IoT devices, or OS functions via standardized MCP servers.

Instead of operating in isolation, the AI gains real-time awareness, enabling more natural and proactive assistance. Collaborating Agents (Agent Societies) - I'm very excited about this one - MCP could also serve as a shared workspace for multi-agent systems. Specialized AI agents - one for research, one for planning, another for execution - could use MCP to exchange information and coordinate tasks dynamically. With MCP, each agent doesn't need direct integrations; they simply access a common toolset. Personal AI Assistants with Deep Integration MCP could let users configure their own AI to interact with personal data and apps securely. A local MCP server could grant an AI access to emails, notes, and smart devices without exposing sensitive data to third parties. This could create an ultra-personalized AI assistant without relying on cloud-based services. Enterprise Governance and Security For businesses, MCP standardizes AI access to internal tools, reducing integration overhead. It also enables governance: AI interactions can be logged, monitored, and controlled via an oversight layer, preventing unintended actions while maintaining efficiency. These are just the early glimpses of MCP's potential. By enabling fluid, context-aware, multi-step interactions, it moves AI agents closer to true autonomous workflow execution. Concluding Thoughts MCP is rapidly maturing into a powerful standard protocol that turns AI from an isolated "brain" into a versatile "doer." By streamlining how agents connect with external systems, it clears the path for more capable, interactive, and user-friendly AI workflows. Key Upcoming Features (based on the workshop from Mahesh Murag from Anthropic) Remote Servers & OAuth Seamless remote hosting using SSE. Built-in OAuth 2.0 for secure integration (e.g., Slack). Official MCP Registry Centralized discovery and verification of servers. Enterprise-friendly: hosts can run private registries. Well-Known Endpoints Standardized .well-known/mcp files for first-party server discovery. Further Enhancements Streaming support, stateless connections, proactive server behavior, and better name spacing. Each update will make MCP more robust, helping AI agents integrate more deeply into real-world workflows. It's a community-driven effort, so keep an eye on the roadmap, join the discussions, and help shape the future of how AI and software intersect. MCP surged, and we even had to change our editorial schedule for it. This topic just begged to be explained. It felt only natural to cover it after discussing Action in agentic workflows. In the next episode, we will explore Human-AI communication and Human-in-the-Loop (HITL) integration, and then move on to Multi-Agent Collaboration. Stay tuned. Sharing this article helps us grow and reach more people - thank you! Resources to Dive Deeper: Introducing the Model Context Protocol by Anthropic Model Context Protocol documentation and quickstart guide MCP docs Model Context Protocol on GitHub Collection of Servers for MCP on GitHub Building Agents with Model Context Protocol (and especially the part: What's next for MCP) by Mahesh Murag from Anthropic, Workshop @AI Engineering Summit Why MCP Won by swyx from Latent Space GitHub Star History (charts) MCP: Flash in the Pan or Future Standard? by LangChain MCP Guardian on Github Exposing Services with MCP Initial reaction to MCP on reddit Sources from Turing Post #1: Open-endedness and AI Agents - A Path from Generative to Creative AI? #5: Building Blocks of Agentic Systems #9: Does AI Remember? The Role of Memory in Agentic Workflows #10: Does Present-Day GenAI Actually Reason? #11: How Do Agents Plan and Reason? #12: How Do Agents Learn from Their Own Mistakes? The Role of Reflection in AI #13: Action! How AI Agents Execute Tasks with

UI and API Tools Thank you for reading! If you want to receive our articles straight to your inbox, please subscribe here Community loaspra Mar 18 I like to view MCP as a higher lever of abstraction on tool usage. First came langchain with its tool functions, and now we have MCP servers that encapsulate a full feature and usability of a specific framework.At first I thought that MCP were more related with the 'reasoning' part of the agent (well they maybe are bc the way tools are defined on the MCP server --markdown like instructions for each tool and the purpose of the MCP server-- they improve the overall performance of the agent).But the most interesting fact about this is that Agent workflows will became more complex. Then we will improve even more the reasoning part (overall intelligence of the agents), maybe we will see more adversarial style of agents (just like MoE but on a higher level). Interesting times ahead See translation 1 reply 3 3 + Kseniase Article author Mar 21 Totally agree! Mario1982 Mar 19 Thank you for this very interesting article! See translation 1 reply 1 1 + Kseniase Article author Mar 21 You are very welcome taehallm Apr 7 You should definitely write about function calling vs. mcp! See translation 1 reply 1 1 + Kseniase Article author Apr 30 great idea ntzaz Apr 14 This was super helpful. Got to know much about MCP in my first few articles. Looking forward to learn more!! Keep doing the good job in sharing the knowledge. See translation 1 reply 1 1 + Kseniase Article author Apr 30 A2A overview coming soon! See translation Boeff Apr 24 Very useful level of detail. It seems the major development is that the choice of tools has been moved from a hard-coded developer decision to a dynamic AI decision. Over the history of IDE development people kept discovering they had a computer available: "Oh wait, we have a computer available, we can do syntax highlighting" and then "Oh wait, we have a computer available we can do pre-compile code validation". Now we are in the stage of "Oh wait, we have an AI available". PS It seems the reference to Mahesh Murthy is an error. See translation 1 reply 1 1 + Kseniase Article author Apr 30 It seems to work for me, check https://www.youtube.com/watch?v=kQmXtrmQ5Zg See translation dbur Apr 30 Thank you for the insightful overview of MCP. I understand that MCP serves as a standardized integration layer for AI agents to perform actions involving external data or tools. Do you think entire agents themselves be served through MCP, or is MCP primarily designed for integrating individual tools and actions within an agent's workflow? See translation 1 reply 1 1 + Kseniase Article author Apr 30 So far I think it's mostly for integrating tools. A2A is about agents and their communication. I will post a detailed overview of A2A here on Hugging face soon See translation 1 1 + harsid21 Jun 2 "Anthropic's Mahesh Murthy delivered a workshop that went viral"I am not sure the name is right. Isnt it Mahesh Murag? This is the workshop right? - https://www.youtube.com/watch?v=kQmXtrmQ5Zg See translation 1 reply Kseniase Article author Jun 2 Thank you! Fixed (don't know how that happened, his name is correct in other places in the article) See translation 1 1 + EditPreview Upload images, audio, and videos by dragging in the text input, pasting, or clicking here. Tap or paste here to upload images Comment Sign up or log in to comment Upvote 313 +301