Tutorial: Large Language Models and AI Agent Systems

Doctoral Seminar 2025

Yongli Mou

Chair of Computer Science 5

Databases and Information Systems

RWTH Aachen University

Aachen, Germany — 26.11.2025

Fraunhofer FIT

Computer Science 5 - Information Systems and Databases | RWTH AACHEN UNIVERSITY

Outline

Part 1: Large Language Models

- What are LLMs
- Transformers
- Pre-Training & Fine-tuning
- Mainstream LLMs
- Capabilities & Limitations

Part 2: AI Agent

- Agent Architecture
- Reasoning
- RAG
- Tools & MCP

Part 3: Practices

- Toy LLM
- Toy MCP
- Toy Agent

Boom of Large Language Models

- **2017:** Transformer architecture introduced [1]
- **2018:** GPT-1, BERT [2]
- **2019–2020:** GPT-2, GPT-3 [3]
- **2022:** ChatGPT ignites the AI boom [4]
- **2023–2025:** GPT-4, Claude [5], Gemini [6], DeepSeek [7], Qwen [8], and other multimodal models.

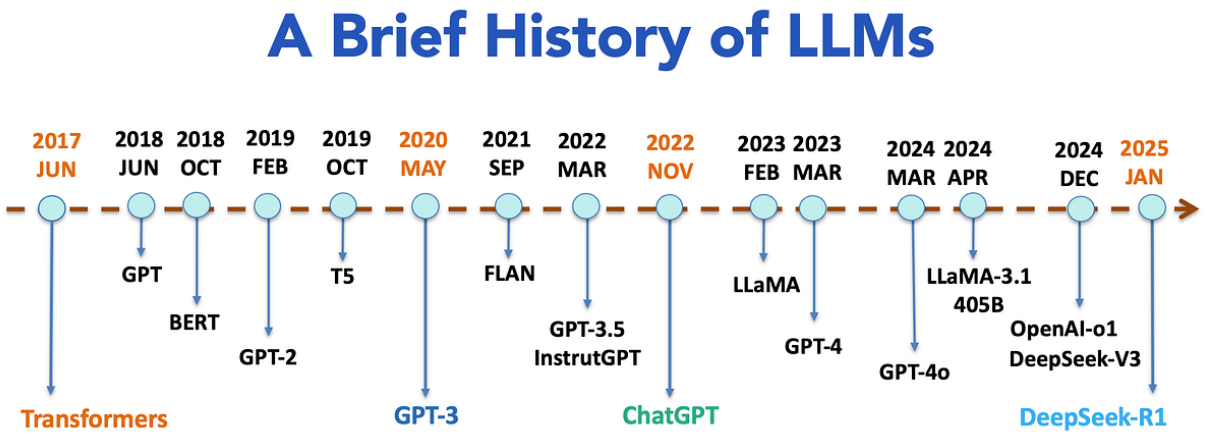


Figure: Timeline of notable LLM releases from Transformer (2017) to recent multimodal systems.

What are Large Language Models

Definition: Large Language Model

A Large Language Model (LLM) is a parameterized conditional probability model, mapping any finite token sequence (context) to a probability distribution over the next token in the vocabulary $\mathcal{V}$. The parameters θ are learned by maximizing log-likelihood over large-scale corpora [9].

$$f_\theta : \mathcal{V}^* \to \Delta(\mathcal{V}),$$

Mathematical Notation

- $\mathcal{V}$: token vocabulary (finite set of discrete symbols)
- $\mathcal{V}^*$: the set of all finite-length sequences over $\mathcal{V}$ (e.g., any prefix $w_{<t}$)
- $\Delta(\mathcal{V})$: the probability simplex over $\mathcal{V}$

$$\Delta(\mathcal{V}) = \{p : \mathcal{V} \to [0,1] \mid \sum_{v \in \mathcal{V}} p(v) = 1\}$$

- Induced joint distribution:

$$P_\theta(w_{1:n}) = \prod_{t=1}^{n} f_\theta(w_t \mid w_{<t})$$

Language Modeling

Language Modeling

Language modeling estimates the joint distribution over a sequence $w_{1:n} = (w_1, \ldots, w_n)$ by autoregressive factorization:

$$P_\theta(w_{1:n}) = \prod_{i=1}^{n} P_\theta(w_i \mid w_{<i}).$$

Training Objective

Maximum likelihood (cross-entropy minimization):

$$\mathcal{L}(\theta) = -\sum_{i=1}^{n} \log P_\theta(w_i \mid w_{<i}) \quad \Longleftrightarrow \quad \min_\theta \ \mathrm{CE}(P_{\text{data}}, P_\theta).$$

Language Modeling Metrics

- **Autoregressive generation:** The model predicts the next token given a prefix:

$$w_{t+1} \sim P_\theta(\cdot \mid w_{\leq t}),$$

- **Context window:** The model conditions only on the most recent L tokens:

$$P_\theta(w_t \mid w_{<t}) \approx P_\theta(w_t \mid w_{t-L:t-1}).$$

- **Perplexity (PPL):** For a sequence $w_{1:n}$, the average negative log-likelihood is

$$\mathcal{L}(\theta) = -\frac{1}{n} \sum_{t=1}^{n} \log P_\theta(w_t \mid w_{<t}).$$

  Perplexity is defined as:

$$\mathrm{PPL} = \exp\Big(\mathcal{L}(\theta)\Big) = \exp\left(-\frac{1}{n} \sum_{t=1}^{n} \log P_\theta(w_t \mid w_{<t})\right).$$

- **Interpretation:** Geometric mean of inverse probabilities. Lower is better.

$$\mathrm{PPL} = \left(\prod_{t=1}^{n} \frac{1}{P_\theta(w_t \mid w_{<t})}\right)^{\frac{1}{n}}.$$

Transformer Architecture

Core Innovation

Transformers replace recurrence with self-attention, enabling parallel token interactions and efficient modeling of long-range dependencies [1].

Layer Structure (Encoder/Decoder block) Given hidden states $H \in \mathbb{R}^{n \times d}$:

$$\tilde{H} = \mathrm{MHA}(H) + H, \qquad H^{+} = \mathrm{FFN}(\tilde{H}) + \tilde{H}.$$

Key Components

- **Self-attention** using learned projections (W_Q, W_K, W_V).
- **Multi-head attention** to attend in multiple subspaces.
- **Positional encoding** to inject sequence order.
- **Position-wise feed-forward network** for nonlinearity and mixing.
- **Residual connections + LayerNorm** for stable optimization.

Self-Attention Mechanism

Construction: For hidden states $H \in \mathbb{R}^{n \times d}$,

$$Q = HW_Q, \quad K = HW_K, \quad V = HW_V,$$

with projection matrices $W_Q, W_K, W_V \in \mathbb{R}^{d \times d}$. In multi-head attention, Q, K, V are reshaped into h heads.

Scaled Dot-Product Attention [1]

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V.$$

Multi-Head Attention (MHA):

$$\text{MHA}(H) = \text{Concat}(\text{head}_1, \ldots, \text{head}_H) W_O.$$

Decoding Strategies in Large Language Models

Greedy Decoding

$$w_{t+1} = \arg\max_{w} P_\theta(w \mid w_{\leq t})$$

Deterministic but often repetitive.

Temperature Sampling Rescales logits before softmax:

$$P_T(w_i) = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

$T < 1$: sharper; $T > 1$: more random.

Top-k Sampling Sample only from the k most probable tokens.

Top-p (Nucleus) Sampling Use smallest set V_p covering probability mass p [10]:

$$\sum_{i \in V_p} P(w_i) \geq p.$$

Adaptive truncation prevents tail degeneration.

Beam Search Maintains B candidate sequences:

$$\max_{w_{1:T}} \sum_{t=1}^{T} \log P_\theta(w_t \mid w_{<t})$$

Effective for translation; less suitable for open-ended chat.

Temperature: Effect on Softmax Distribution

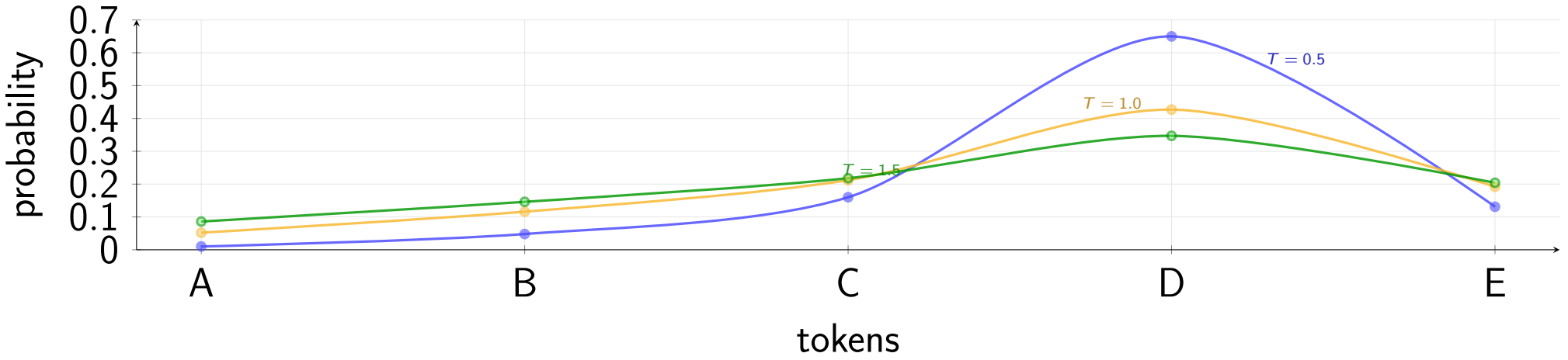
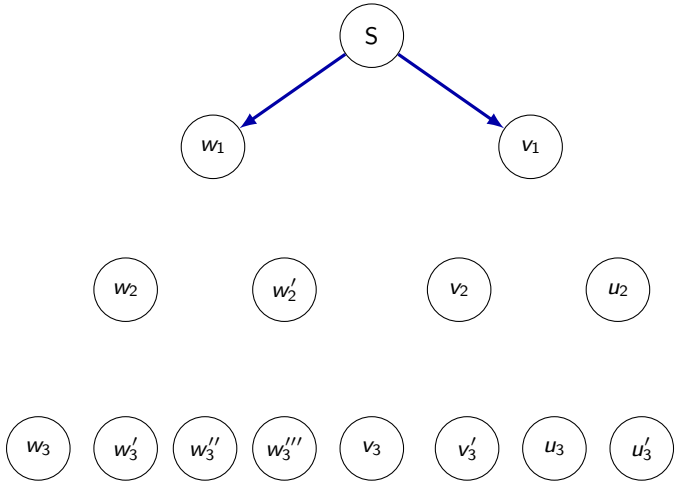


Figure: Temperature scaling reshapes the same logits.

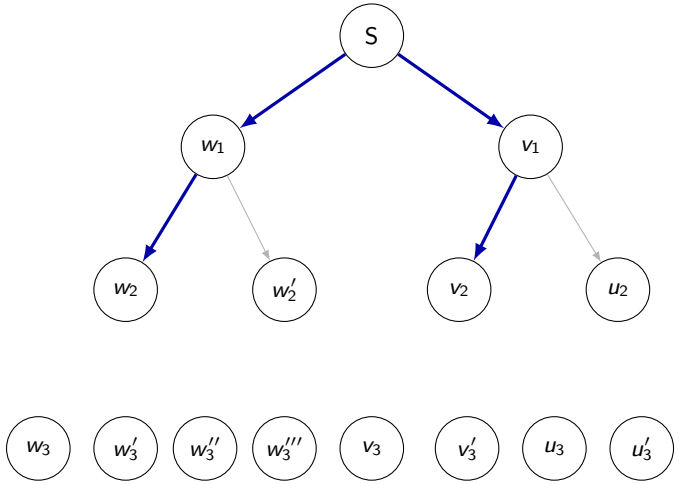
Beam Search (Beam Width $B = 2$)



Step 1: keep top-2 beams

Figure: Beam width $B = 2$: expand all candidates each step, keep the two highest-scoring prefixes (blue).

Beam Search (Beam Width $B = 2$)

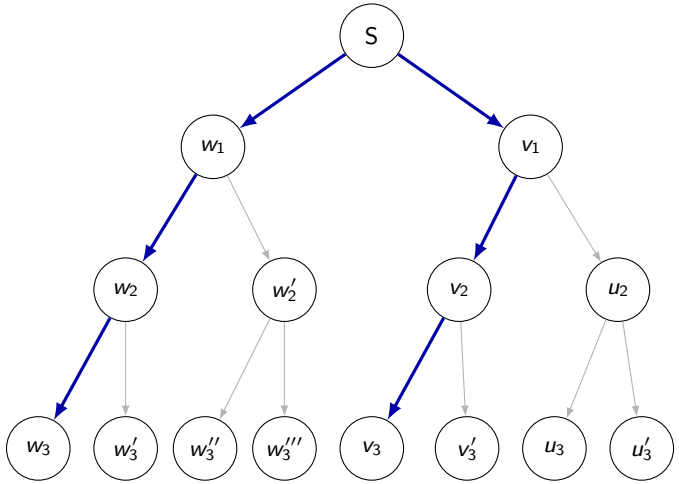


Step 1: keep top-2 beams

Step 2: re-score, keep new top-2

Figure: Beam width $B = 2$: expand all candidates each step, keep the two highest-scoring prefixes (blue).

Beam Search (Beam Width $B = 2$)



Step 1: keep top-2 beams

Step 2: re-score, keep new top-2

Step 3: repeat until stop

Figure: Beam width $B = 2$: expand all candidates each step, keep the two highest-scoring prefixes (blue).

Training Paradigm Evolution

From Pretraining to Alignment

- **Pretraining:** large-scale next-token prediction [3].

$$\min_{\theta} -\sum_{t} \log P_\theta(w_t \mid w_{<t})$$

  Foundation model learns general linguistic and world knowledge.

- **Fine-tuning (general concept):** adapt pretrained parameters θ_0 to a new distribution:

$$\theta^* = \arg\max_{\theta} \sum_{(x,y)\in\mathcal{D}_{ft}} \log P_\theta(y \mid x)$$

  Includes SFT, RLHF, RLAIF, DPO [11], etc.

Modern alignment pipeline:

$$\text{Pretraining} \rightarrow \text{SFT} \rightarrow \text{Preference Optimization (RLHF / RLAIF)}$$

Modern Alignment Techniques

Supervised Fine-Tuning (SFT)

- Dataset of instruction–response pairs $\mathcal{D}_{SFT} = \{(x_i, y_i)\}$.
- Optimize cross-entropy (teacher forcing) [4].
- Teaches format, compliance, basic helpfulness.

Reinforcement Learning from Human Feedback (RLHF)

- **Reward Model (RM)** trained on preference pairs [12].
- **PPO optimization** with KL constraint:

$$\max_{\theta} \mathbb{E}[\, r_\phi(x, y)\,] - \beta\, \mathrm{KL}(\pi_\theta \| \pi_{SFT})$$

- Improves helpfulness, harmlessness, and alignment.

RLAIF and Scalable Oversight

Reinforcement Learning from AI Feedback (RLAIF)

- Replace human preference labels with a strong AI judge.
- Construct AI preference pairs $(x, y^+, y^-)_{\text{AI}}$ [13].
- Enables scalable, cheaper, principle-based alignment (Constitutional AI).

Training Paradigm Summary

- **SFT**: learns to follow instructions.
- **RLHF**: aligns to human preferences.
- **RLAIF**: aligns via AI-based preference judgments.

$$\text{SFT: “can talk”} \quad \Rightarrow \quad \text{RLHF/RLAIF: “talks well”}$$

Major LLM Families

Decoder-Only Models (Autoregressive)

- GPT Series (OpenAI) [3]
- LLaMA (Meta) [14]
- Qwen (Alibaba) [8]
- Mistral / Mixtral (MoE) [15]
- DeepSeek [7]

Encoder-Only Models (Bidirectional)

- BERT (Google) [2]
- RoBERTa, ALBERT
- Focused on understanding tasks

Encoder–Decoder Models (Seq2Seq)

- T5 / FLAN-T5 [16]
- Strong for translation, summarization

Multimodal Models

- GPT-4 / GPT-4o (OpenAI)
- Gemini (Google) [6]
- Claude 3 (Anthropic) [5]

Notable Architectures and Innovations

Open-Source Ecosystem:

- **LLaMA**: flexible scales, strong baseline for fine-tuning.
- **Mistral/Mixtral**: efficient attention, MoE routing [15].
- **DeepSeek**: optimized training pipeline and inference efficiency [7].

Key Innovations:

- **MoE (Mixture of Experts):** Activates only a subset of parameters per token (Mistral, DeepSeek).
- **Long Context:** Handling 1M+ tokens (Gemini, Claude).
- **Attention Efficiency:** FlashAttention, RoPE, ALiBi.

Capabilities of Large Language Models

Core Strengths of LLMs [9]

- **Language Generation:** Produce coherent, context-aware text.
- **Semantic Understanding:** Encode rich contextual representations.
- **In-Context Learning:** Adapt to new tasks from examples without updates [3].
- **Reasoning Heuristics:** Chain-of-thought, code synthesis.
- **Multimodal Integration:** Align text, image, audio.
- **Knowledge Compression:** Store world knowledge in parameters.

LLMs are powerful *latent knowledge engines*.

Limitations & Motivation for AI Agents

Fundamental Limitations [17]

- **Lack of Groundedness:** No direct access to real-time information.
- **No Persistent Memory:** Context is bounded.
- **Unreliable Reasoning:** Hallucinations.
- **Static Knowledge:** Knowledge is "frozen" at training time.
- **Weak Action Capabilities:** Cannot execute actions reliably.

Why AI Agents?

- Integrate LLMs with **tools**, **memory**, **environment** [18].
- Enable planning, real-world actions, continual information gathering.
- Transform LLMs from passive text generators to **autonomous problem-solvers**.

What Are AI Agents?

Definition

An **AI Agent** is a system that uses an LLM as its core controller to **perceive**, **reason**, and **act** within an external environment [18]. Formally, an agent defines a policy

$$\pi : \mathcal{O}^* \to \mathcal{A},$$

mapping a history of observations $\mathcal{O}^*$ to actions $\mathcal{A}$ [17].

Key Characteristics

- **Autonomy:** Operates in iterative loops (The "Agentic Loop") without continuous human intervention.
- **Goal-driven:** Executes high-level objectives rather than just responding to prompts [19].
- **Tool Usage:** Interacts with external APIs (Search, Calculator, Code Interpreter).

Paradigm Shift: Copilot vs. Agent

Copilot (Human-in-the-loop)

- **Role:** Assistant / Draftsman.
- **Trigger:** User explicitly prompts every step.
- **Flow:** Linear (Input $\rightarrow$ Output).
- *Example:* GitHub Copilot, ChatGPT.

Agent (Human-on-the-loop)

- **Role:** Actor / Executor.
- **Trigger:** User sets a high-level goal.
- **Flow:** Looping (Thought $\rightarrow$ Action $\rightarrow$ Observation).
- *Example:* AutoGPT, Devin.

The Cognitive Architecture

An Agent is not just a model; it is a system designed to act autonomously. Based on Lilian Weng's architecture [18]:

Core Components:

1. **Brain (LLM):** The core controller for reasoning and planning.
2. **Memory:** Stores history and knowledge.
3. **Planning:** Task decomposition and self-reflection.
4. **Tools:** Executing actions and manipulating environments.

Memory Systems in Agents

Inspired by human cognitive science, Agent memory is categorized into three types [19]:

1. **Working Memory (Short-term):**
   - *Implementation:* The Context Window.
   - *Function:* Stores current reasoning steps, immediate observations, and scratchpad data.
2. **Episodic Memory (Experience):**
   - *Implementation:* Vector Database (Logs/Trace).
   - *Function:* Recalling past events, user interactions, and outcomes of previous actions to learn from experience.
3. **Semantic Memory (Knowledge):**
   - *Implementation:* Model Weights + RAG (Knowledge Base).
   - *Function:* Storing facts about the world (e.g., "The capital of France") independent of personal experience.

Reasoning: Beyond Simple Prompting

Agents require structured thinking to solve complex, multi-step problems.

- **Chain of Thought (CoT):** *"Let's think step by step."* Decomposing a problem into linear intermediate steps [20].
- **Tree of Thoughts (ToT):** Generalizes CoT by exploring multiple "branches" of reasoning possibilities. The agent can look ahead, backtrack, and evaluate different paths globally [21].
- **Reflexion (Self-Correction):** An architecture where the agent critiques its own past failures. It generates a "verbal reinforcement" trace to avoid repeating mistakes in the next attempt [22].

RAG: The Agent's Library

RAG bridges the gap between the frozen parametric knowledge of the LLM and dynamic external data [23].

Why Agents Need RAG?

- **Hallucination Reduction:** Grounds answers in retrieved documents.
- **Dynamic Knowledge:** Access to up-to-date information (e.g., stock prices, recent news) without retraining.
- **Domain Specificity:** Access to private enterprise data.

The Flow:

$$\text{Query} \xrightarrow{\text{Embed}} \text{Vector Search} \xrightarrow{\text{Retrieve}} \xrightarrow{\text{Augment Prompt}} \text{LLM}$$

Tools & Action Execution

Agents use tools to affect the world, turning text output into executable actions [24].

The Tool Execution Loop

1. **Select:** LLM decides which tool to call based on the prompt.
2. **Generate:** LLM formats valid JSON arguments (e.g., ‘”calc”: ”12*4”‘).
3. **Execute:** The runtime (Python/API) runs the function.
4. **Observe:** Result feeds back into Working Memory.

The Integration Challenge

Without a standard, connecting Agents to data is an $O(M \times N)$ problem.

- **The Silo Issue:** Developers must write specific integration code for every data source (Google Drive, Slack, GitHub) for every different Agent implementation [25].
- **Fragmentation:** An agent built for Claude Desktop cannot easily access tools built for a LangChain application.
- **Maintenance Nightmare:** API changes in a data source break all connected agents.

Enter MCP: The "USB-C" for AI

Model Context Protocol (MCP) is an open standard that unifies how AI models interact with data and tools [26].

Core Philosophy: "Build once, run anywhere."

MCP Architecture [27]

- **MCP Host:** The application where the AI model runs (e.g., Claude Desktop, IDEs like Cursor/Windsurf). It manages the connection lifecycle.
- **MCP Client:** The protocol implementation within the Host. It maintains 1:1 connections with Servers and handles permission negotiation.
- **MCP Server:** A lightweight bridge to a specific data source. It exposes capabilities (Resources, Tools) to the Client without knowing the implementation details of the Host.

MCP Primitives: What Servers Expose

MCP Servers expose three primary primitives to the Agent [27]:

1. **Resources (Passive Context):**
   - *Role:* Reading data (File content, database rows, logs).
   - *Analogy:* Like ‘GET‘ requests; providing the "ground truth".
2. **Tools (Active Capabilities):**
   - *Role:* Executing actions (API calls, scripts).
   - *Analogy:* Like ‘POST‘ requests; model-controlled execution.
3. **Prompts (Reusable Workflows):**
   - *Role:* Pre-defined templates to help users leverage the server.
   - *Example:* A "Debug Error" prompt that automatically loads the error log Resource and the code context.

Building AI Agent Systems

Hands-on focus

From a toy Transformer to a chat agent with tools and a runnable UI stack.

- **LLM build and pretrain** — craft a minimal Transformer in `torch`, wire a tiny dataset, and pretrain/inspect with `deepspeed`.
- **MCP server & client** — define resources and tools, ship prompts, and exercise them via a toy command-line client.
- **Agent workflow** — assemble a chat agent that hits the MCP tools, with a lightweight frontend, backend, and workflow engine.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of NAACL-HLT*. The BERT paper. 2019.

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[4] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. "Training language models to follow instructions with human feedback". In: *Advances in neural information processing systems* 35 (2022), pp. 27730–27744.

[5] Anthropic. "The Claude 3 Model Family: Opus, Sonnet, Haiku". In: *Anthropic Technical Report* (2024).

[6] G. Team, R. Anil, S. Borgeaud, Y. Wu, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, et al. "Gemini: a family of highly capable multimodal models". In: *arXiv preprint arXiv:2312.11805* (2023).

[7] DeepSeek-AI. "DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model". In: *arXiv preprint arXiv:2405.04434* (2024).

[8] J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng, Y. Fan, W. Ge, Y. Ge, G. Han, et al. "Qwen technical report". In: *arXiv preprint arXiv:2309.16609* (2023).

[9] H. Naveed, A. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes, and A. Mian. "A comprehensive overview of large language models". In: *ACM Transactions on Intelligent Systems and Technology* 16.5 (Aug. 2025), pp. 1–72.

[10] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi. "The curious case of neural text degeneration". In: *International Conference on Learning Representations*. Introduced Nucleus (Top-p) Sampling. 2020.

[11] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn. "Direct preference optimization: Your language model is secretly a reward model". In: *Advances in Neural Information Processing Systems*. Vol. 36. DPO paper. 2024.

[12] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. "Deep reinforcement learning from human preferences". In: *Advances in Neural Information Processing Systems*. Vol. 30. Foundational RLHF paper. 2017.

[13] Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, et al. "Constitutional AI: Harmlessness from AI Feedback". In: *arXiv preprint arXiv:2212.08073* (2022). The key paper for RLAIF and Constitutional AI.

[14] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. "Llama 2: Open foundation and fine-tuned chat models". In: *arXiv preprint arXiv:2307.09288* (2023).

[15] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, et al. "Mistral 7B". In: *arXiv preprint arXiv:2310.06825* (2023).

[16] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. "Exploring the limits of transfer learning with a unified text-to-text transformer". In: *The Journal of Machine Learning Research* 21.1 (2020). The T5 paper, pp. 5485–5551.

[17] R. Sapkota, K. I. Roumeliotis, and M. Karkee. "AI agents vs. agentic AI: A conceptual taxonomy, applications and challenges". In: *arXiv preprint arXiv:2505.10468* (2025).

[18] L. Weng. *LLM Powered Autonomous Agents*. lilianweng.github.io. Key reference for Agent Architecture (Memory, Planning, Tools). June 2023. URL: `https://lilianweng.github.io/posts/2023-06-23-agent/`.

References

[19] J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein. "Generative agents: Interactive simulacra of human behavior". In: *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 2023, pp. 1–22.

[20] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. "Chain-of-thought prompting elicits reasoning in large language models". In: *Advances in neural information processing systems* 35 (2022), pp. 24824–24837.

[21] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan. "Tree of thoughts: Deliberate problem solving with large language models". In: *Advances in Neural Information Processing Systems*. Vol. 36. Tree of Thoughts (ToT). 2023.

[22] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao. "Reflexion: Language agents with verbal reinforcement learning". In: *Advances in Neural Information Processing Systems*. Vol. 36. Reflexion: Self-correction mechanism. 2023.

[23] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. "Retrieval-augmented generation for knowledge-intensive nlp tasks". In: *Advances in Neural Information Processing Systems*. Vol. 33. The foundational RAG paper. 2020, pp. 9459–9474.

[24] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom. "Toolformer: Language models can teach themselves to use tools". In: *Advances in Neural Information Processing Systems* 36 (2024).

[25] H. Chase. "LangChain: Building applications with LLMs through composability". In: GitHub. 2022.

[26] Anthropic. *Model Context Protocol (MCP)*. `https://modelcontextprotocol.io/`. The open standard for connecting AI models to data and tools. 2024.

[27] Model Context Protocol. *Model Context Protocol Specifications*. `https://github.com/modelcontextprotocol`. 2024.