CSAI 422 - Applied Generative AI

LLMs and Prompting

DATE : 05/09/2022

Course Structure

- Prompt Engineering for LLMS
- Retrieval Argument Generation
- Agentic Systems
- Stable Diffusion and Image/Video Generation

Learning Outcomes

By the end of this lecture you will know about:

- The 5th Industrial Revolution, the future or work
- Is this the end of programming?
- What are LLMs?
- What is Prompt Engineering

Industrial Revolutions

Industrial Revolutions

- 1st Industrial Revolution (1760s–1830s): Mechanization & Steam Power
  - **Key Technologies**: Steam engines, mechanized textile manufacturing, railroads.
  - **Impact**: Transition from agrarian societies to industrial economies. Factories replaced small-scale artisans, leading to urbanization.
  - **Labor Shift**: Manual laborers replaced by machines in textile, mining, and agriculture.

Industrial Revolutions

- 2nd Industrial Revolution (1870s–1914): Electrification & Mass Production
  - **Key Technologies**: Electricity, internal combustion engines, steel production, chemical industries.
  - **Impact**: Introduction of assembly lines (Ford's Model T), mass production, and new industries like automobiles and telecommunications.
  - **Labor Shift**: Rise of blue-collar factory jobs; emergence of consumer goods on a large scale.

Industrial Revolutions

3rd Industrial Revolution (1950s–2000s): Computers & Automation

- **Key Technologies**: Transistors, microprocessors, personal computers, internet, early robotics.
- **Impact**: Shift from mechanical to digital technology; automation of repetitive tasks in manufacturing and business.
- **Labor Shift**: Decline in manual jobs due to automation, rise of knowledge-based work in computing and IT.

Industrial Revolutions

3rd Industrial Revolution (1950s–2000s): Computers & Automation

- **Key Technologies**: Transistors, microprocessors, personal computers, internet, early robotics.
- **Impact**: Shift from mechanical to digital technology; automation of repetitive tasks in manufacturing and business.
- **Labor Shift**: Decline in manual jobs due to automation, rise of knowledge-based work in computing and IT.

Industrial Revolutions

4th Industrial Revolution (2000s–Present): AI, IoT & Cyber-Physical Systems (Industry 4.0)

- **Key Technologies**: Artificial intelligence (AI), cloud computing, Internet of Things (IoT), blockchain, autonomous systems.
- **Impact**: Smart factories, hyper-automation, machine learning-driven decision-making, digital twins.
- **Labor Shift**: AI replacing white-collar tasks (customer support, coding, content creation), new roles in AI ethics, cybersecurity, and data science.

Industrial Revolutions

5th Industrial Revolution

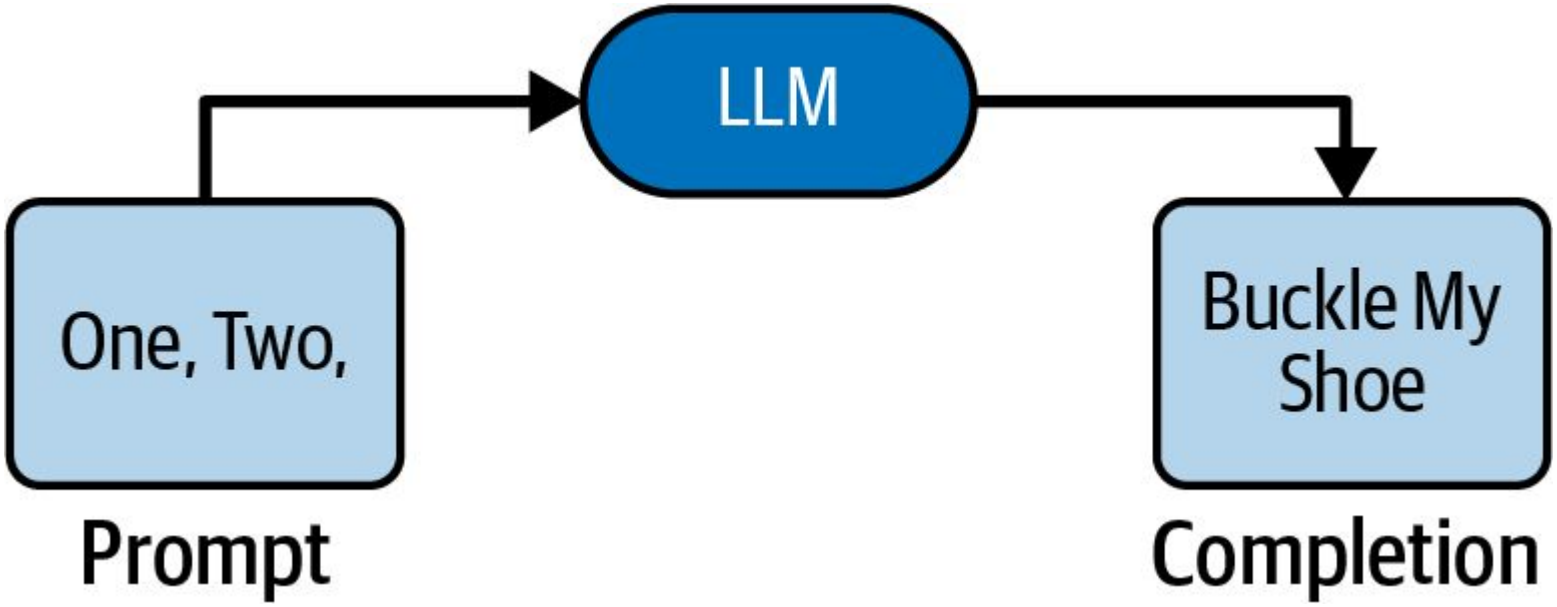
Possible features of Industry 5.0:

- **AI-human co-creation**: AI becomes a creative collaborator, not just an automation tool.
- **Personalization at scale**: AI-driven products tailored to individual preferences.
- **AI ethics and regulation**: Addressing biases, misinformation, and security risks.
- **Sustainable AI**: Energy-efficient models, AI for climate solutions.

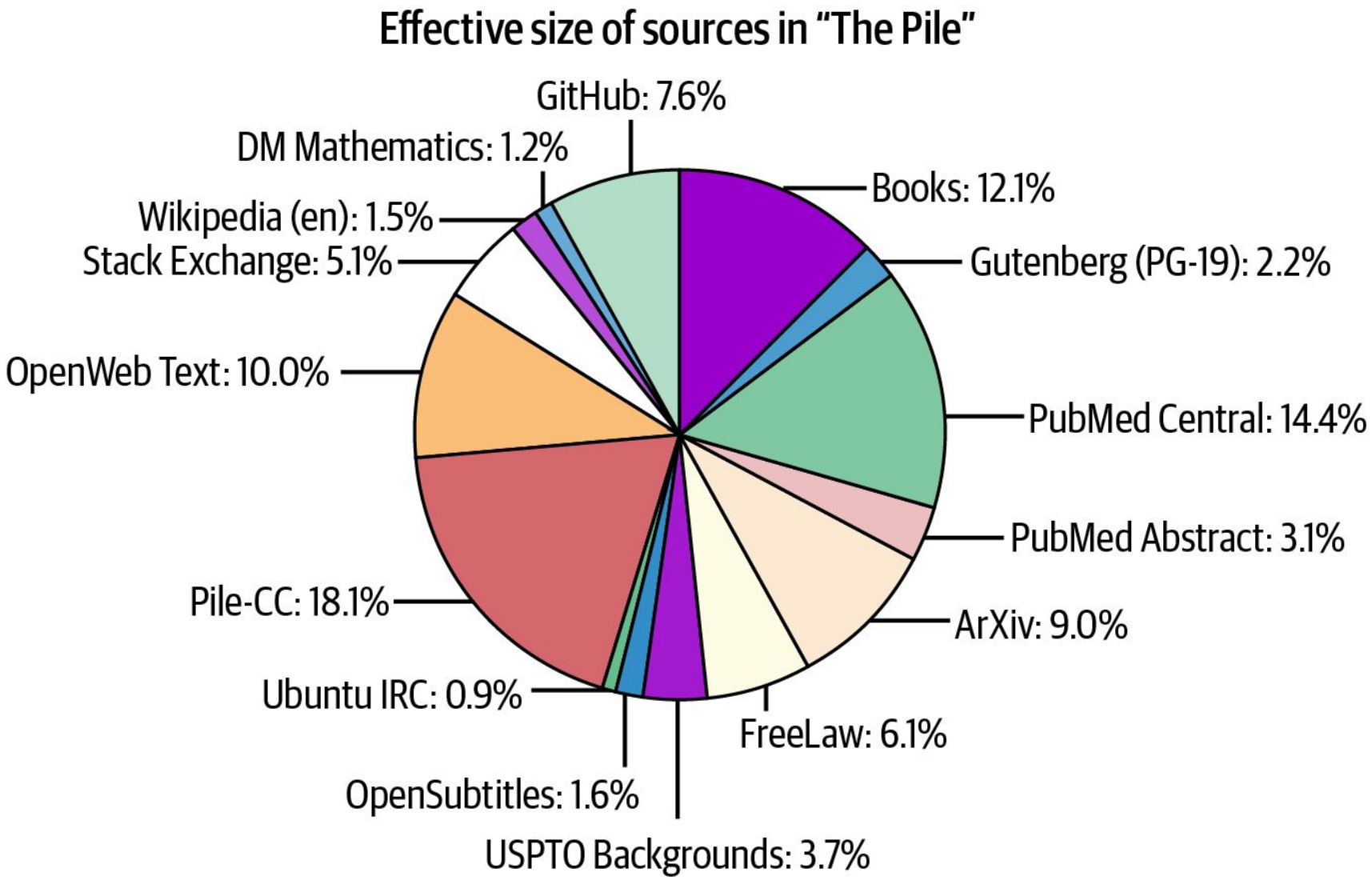
LLMs (what are those!)

LLMs

NEXT WORD PREDICTORS….since the 1940s… markov chains



LLMs


Effective size of sources in "The Pile"
GitHub: 7.6%
DM Mathematics: 1.2%
Books: 12.1%
Wikipedia (en): 1.5%
Gutenberg (PG-19): 2.2%
Stack Exchange: 5.1%
OpenWeb Text: 10.0%
PubMed Central: 14.4%
PubMed Abstract: 3.1%
Pile-CC: 18.1%
ArXiv: 9.0%
Ubuntu IRC: 0.9%
FreeLaw: 6.1%
OpenSubtitles: 1.6%
USPTO Backgrounds: 3.7%

LLMs


```
Yesterday, my TV stopped working. Now, I can't turn it on at
```

For a text that starts like this, what might be the statistically most likely completion?

1. `y2ior3w`
2. `Thursday.`
3. `all.`

The Future of programming

Historical progression

- **Early Programming**:
  - The first programmers connected physical circuits to perform calculations.

- **Machine Code Era**:
  - Programmers wrote machine instructions in binary code.
  - Input was done one bit at a time by flipping switches on a computer's front panel.
- **Introduction of Assembly Language**:
  - Replaced binary machine code.
  - Allowed programmers to use a human-like language to move data and perform calculations.
- **Higher-Level Compiled Languages**:
  - Fortran, COBOL, and later C, C++, and Java reduced the need for assembly programming.
  - Enabled programmers to use higher-level abstractions to communicate with computers.

Programmer as Manager

The workers here are the programs and the computers… not the developers!

The Developer of the New Age

- **Chat-Oriented Programming (CHOP)**:
    - Nonprogrammers can now interact with LLMs or AI agents in natural language.
    - AI can generate working prototypes in Python or other programming languages.
- **Rise of Advanced AI Models**:
    - AI can now generate complex programs from high-level prompts.
    - Some predict AI will replace human programmers and knowledge workers, leading to mass unemployment.
- **Skepticism Toward Complete AI Replacement**:
    - Technology breakthroughs empower more people but also create new demands.
    - New innovations lead to specialized knowledge that only a few understand.
- **Opportunities for Developers**:
    - AI-driven advancements enable a new era of creativity and exploration.
    - Smart developers who embrace AI will be in demand for their ability to leverage it effectively.

The Transformation

- **AI Will Transform Programming**:
    - AI will not replace programmers but will significantly change their roles.
    - Many current programming tasks may become obsolete, except in specialized fields like embedded systems.
- **Impact on Developers**:
    - Steve Yegge predicts that those who resist new tools and paradigms will be replaced, not junior or mid-level programmers.
    - Developers who embrace AI-driven tools will be in high demand.
- **Shifting Skill Dynamics**:
    - Junior developers proficient in AI tools can outperform senior programmers who fail to adapt.
    - Yegge refers to this shift as **“The Death of the Stubborn Developer.”**

Reflecting on the Past

- **Parallels Between AI and the Industrial Revolution**:
    - AI's impact on programming mirrors the shift during the first Industrial Revolution.
    - Skilled crafters in textile mills were replaced by machines operated by "unskilled" labor.
- **Observations from Economic History**:
    - James Bessen studied wage records from 1800s textile mills in Lowell, Massachusetts.
    - While traditional artisans lost their roles, factory workers developed new skills over time.
- **Skill Evolution, Not Elimination**:
    - Despite automation, workers in both eras took a similar amount of time to reach full proficiency.
    - AI will not eliminate programmers but will require them to develop different skills, just as industrial workers did.

Reflecting on the Past

- **Parallels to the Industrial Revolution**:
    - Wages remained flat or depressed for the first 50 years due to:
        - Factory owners hoarding productivity benefits.
        - Slow adoption and refinement of new technologies.
- **Challenges of New Technological Adoption**:
    - Productivity gains took decades because:
        - Machines needed to be improved and made more robust.
        - New workflows and products had to be developed.
        - Businesses needed time to integrate new technologies.
        - Workers had to acquire skills not just to use but also to repair and innovate.
- **The Process of 'Learning by Doing'**:
    - A technological revolution requires a stable, trained workforce, not just a few skilled individuals.
    - Widespread AI literacy is necessary for companies and industries to thrive.
- **Programming as Human-Machine Communication**:
    - Programming is evolving into a more human-language-like interaction.
    - AI enables machines to understand humans rather than requiring humans to speak in code.
- **The Future of Work in AI-Driven Industries**:
    - AI will create more demand for programs, leading to new industries.
    - Automation often increases employment as demand rises.
    - We are still far from reaching the saturation point where automation reduces job opportunities in programming.

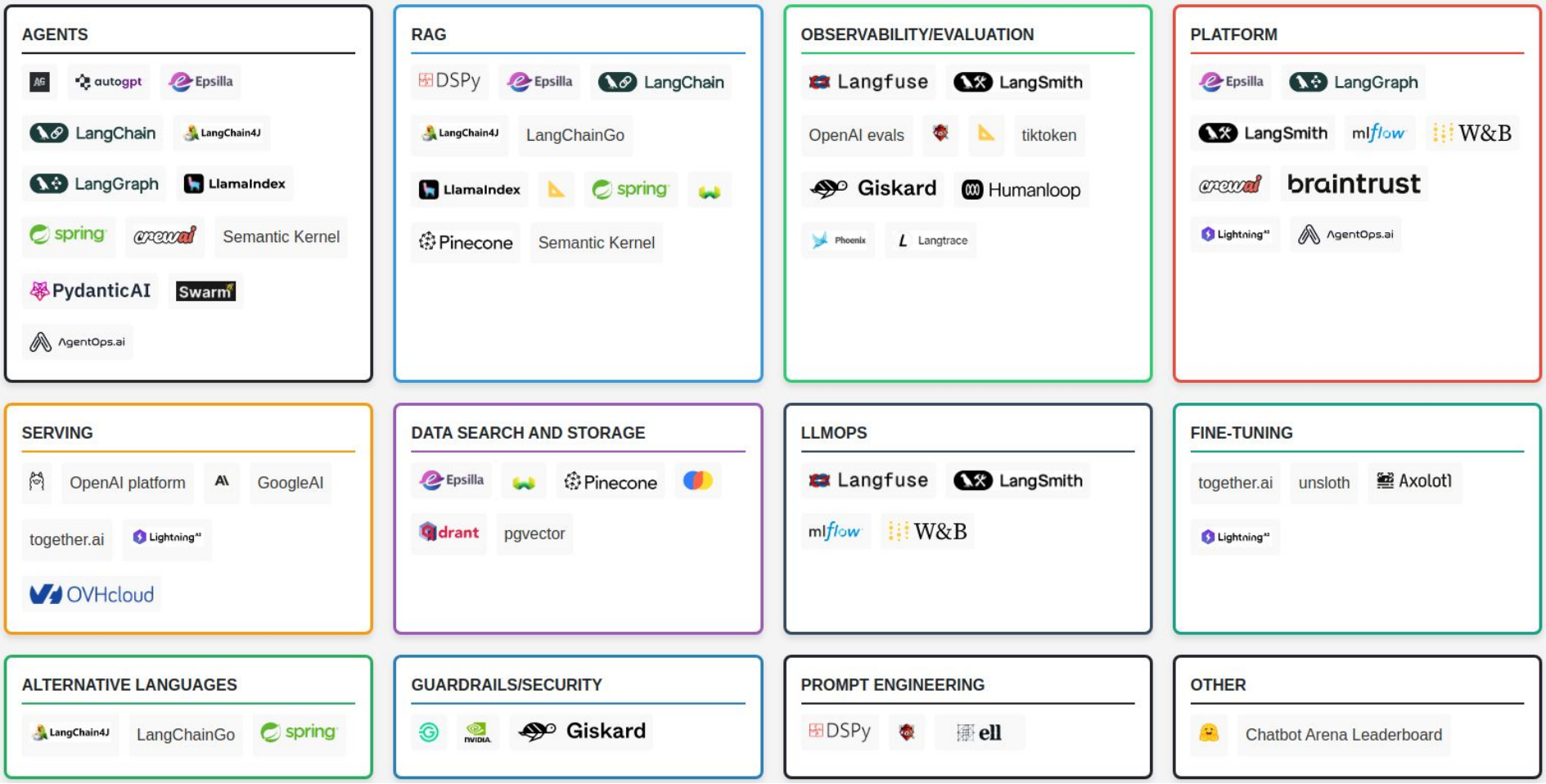
AI as an Empowerment Tool, Not a Replacement

- **Ethan Mollick's Perspective on AI Adoption**:
  - Wharton professor and AI advocate Ethan Mollick supports Bessen's insights.
  - Advocates for **"always bringing AI to the table"** in every aspect of work.
  - Encourages exploring **"the jagged edge"**—understanding both AI's strengths and limitations.
- **AI as an Empowerment Tool, Not a Replacement**:
  - Companies should focus on using AI to enhance workers' abilities rather than replace them.
  - The biggest opportunities lie in **learning how to apply AI effectively**.
- **Workforce-Driven AI Innovation**:
  - Businesses should treat employees' AI experimentation as **applied R&D**.
  - Employees using AI to solve problems will uncover **new solutions and opportunities**.

AI: The Next Paradigm Shift in Programming

- **AI as a New Programming Paradigm**:
  - Sam Schillace, Microsoft Deputy CTO, agrees that AI is reshaping programming.
  - We are in the midst of **inventing a new programming paradigm** centered around AI systems.
- **Historical Shift in Programming Paradigms**:
  - The transition from **desktop to internet** changed everything in the tech stack.
  - Despite keeping the same fundamental layers, their implementations evolved:
    - **Languages**: Shifted from compiled to interpreted.
    - **Development processes**: Moved from Waterfall → Agile → CI/CD.
    - **Databases**: Transitioned from **ACID (traditional)** → **NoSQL (scalable, flexible)**.
    - **Computing model**: From **single-user, single-threaded apps** → **distributed, multi-user systems**.
- **Parallel to AI's Current Transformation**:
  - AI is driving a similar **fundamental transformation** across programming and software development.
  - Just as past paradigm shifts redefined software engineering, AI is **restructuring tools, workflows, and system architectures**.

Emerging Tech Stack



Meta-Cognition and Automation

- **AI's Limitation in Memory and Context**
  - AI models lack memory in the way humans do.
  - Even with large context windows, they struggle with **metacognition**—the ability to reflect on and adapt their own thinking.
  - Humans still need to provide **context** for AI to function effectively.
- **AI as Thought Automation**
  - AI today is at an early stage, akin to the **Industrial Revolution's automation of motion**.
  - Current AI tasks—**summarization, pattern recognition, and text generation**—are basic, like early steam engines pumping water.
  - We have not yet reached the **"locomotive stage"** of AI.
- **The Next Breakthrough: Control Over AI Power**
  - Early industrial advances focused on increasing raw power; the key was **developing control mechanisms**.
  - AI may require **entirely new paradigms and disciplines** beyond traditional software engineering.
  - The challenges of AI will drive the emergence of **new sciences**—cognition, reliability, and scalability.
  - Just as early steam power led to metallurgy, AI will necessitate **new frameworks for understanding and control**.

Moving to Locomotive Stage

The Industrial Revolution Parallel

- In the early days of the Industrial Revolution, machines were crude and inefficient, much like today's AI systems.
- The **first industrial machines** focused on **brute-force automation**—simple, repetitive mechanical tasks such as pumping water or hammering metal.
- The **steam engine** revolutionized industry, but its **true transformation came with locomotives**—not just generating power, but controlling and directing it efficiently.

What the Locomotive Stage Meant for Industry

- The invention of **railroads** was a leap beyond isolated machines:
  - It allowed for the **systematic application of mechanical power** across vast distances.
  - It required **new engineering disciplines**—metallurgy, precision mechanics, and thermodynamics—to make engines reliable and scalable.
  - It **reorganized industries**, creating entire networks of trade, production, and infrastructure.

Where AI is Today

- **Current AI capabilities** resemble the early steam engine phase—performing simple, **brute-force cognitive tasks**:
  - Summarizing text
  - Recognizing patterns
  - Generating responses based on predefined structures
- AI **has power but lacks control**—it can generate vast amounts of content, but struggles with **coherence, long-term reasoning, and self-improvement**.
- Just as industrial engineers had to **invent control systems** for locomotives (such as pressure regulation and precision machining), AI will require **new disciplines** for managing and refining its cognitive power.

Moving to Locomotive Stage

What the AI Locomotive Stage Will Look Like

- **Systematic, controlled, and scalable intelligence**—not just generating information, but reasoning over it.
- **Higher-order metacognition**—AI that can **reflect on its own outputs**, learn dynamically, and improve its decision-making over time.
- **Specialized AI control systems**—akin to how locomotives required **advanced engineering**, AI will demand new fields in:
  - Cognitive reliability (ensuring logical consistency)
  - AI scalability (handling complex, multi-step reasoning)
  - AI-human collaboration (augmenting rather than replacing human intelligence)

The Path Forward

- Just as locomotives turned **steam power into an organized system of controlled motion**, AI needs frameworks to **turn raw computation into structured, adaptive intelligence**.
- The challenge isn't just making AI **more powerful**—it's making it **more precise, interpretable, and useful in real-world applications**.
- We are **building the rails for AI**—the infrastructure that will allow it to become a true **general-purpose cognitive tool**, much like locomotives transformed industry and transportation.

AI Agents: The New Digital Interface

- **AI Agents as Core Business Interfaces**
  - Companies' AI agents will become as significant as websites and mobile apps—potentially even more important.
  - These agents will need to **encode key business policies and processes** to function effectively.
- **Challenges in Implementation**
  - AI may eventually automate its own setup, but today, companies require dedicated engineering teams to integrate AI agents.
  - The **"last mile" challenge**: Translating business processes into AI-driven interactions is complex and requires specialized expertise.
- **Emergence of the Agent Engineer**
  - A new software development role is forming: **Agent Engineer**.
  - Similar to frontend developers, **React developers can transition into AI agent development**.
  - This presents a **valuable reskilling opportunity** for developers to stay relevant in the AI-driven tech landscape.
- **AI Agents and Business Transformation**
  - AI agents will replace traditional **customer service phone trees** with **intelligent, problem-solving interfaces**.
  - However, **success depends on rethinking business processes**, not just replicating existing workflows.
  - An AI agent that simply mimics old systems will be **as ineffective as a digital form that just copies a paper version**.
- **The Real Challenge: Business Process Understanding**
  - The hardest part isn't the programming—it's **deeply understanding business operations** and reimagining them for AI-driven transformation.
  - Companies must focus on leveraging AI's unique capabilities rather than just automating existing workflows.

The 70% problem!

Addy Osmani, the head of user experience for Google Chrome, calls this the 70% problem: "While engineers report being dramatically more productive with AI, the actual software we use daily doesn't seem like it's getting noticeably better." He notes that nonprogrammers working with AI code generation tools can get out a great demo or solve a simple problem, but they get stuck on the last 30% of a complex program because they don't know enough to debug the code and guide the AI to the correct solution.

Experience counts…House of Card Coding

When you watch a senior engineer work with AI tools like Cursor or Copilot, it looks like magic. They can scaffold entire features in minutes, complete with tests and documentation. But watch carefully, and you'll notice something crucial: They're not just accepting what the AI suggests…. ***They're applying years of hard-won engineering wisdom to shape and constrain the AI's output****. The AI is accelerating their implementation, but their expertise is what keeps the code maintainable. Junior engineers often miss these crucial steps.* ***They accept the AI's output more readily, leading to what I call "house of cards code" – it looks complete but collapses under real-world pressure.***

AI and the Changing Definition of Programming

- **AI Doesn't Create New Thinking—It Reveals What Requires Thinking**
  - AI helps distinguish between **mechanical tasks** and **true intellectual work**.
- **Historical Parallel: Writing as an Intellectual Task**
  - In the past, writing was considered an **intellectual skill** because literacy was rare.
  - Calligraphy was once highly valued, but now **writing refers to idea organization, not handwriting**.
- **The Future of Programming**
  - As AI automates coding, **"programming" will shift in meaning**, just as writing did.
  - Programming will focus on **arranging ideas into executable logic**, rather than manually writing code.
- **Computer Science Beyond Coding**
  - **Mehran Sahami (Stanford CS Chair)**:
    - *"Computer science is about systematic thinking, not writing code."*
  - AI highlights that true programming lies in **conceptual and systematic problem-solving**, not just syntax.

Inventing the Future: The Expanding Role of AI in Programming

- **Early Days of Transformation**
  - We are still in the **early stages** of defining how AI will reshape programming and innovation.
  - There is **much to learn and explore** as AI augments software development.
- **AI as a Force Multiplier**
  - AI co-developers could make programmers **10x more productive**.
  - Productivity gains depend on **how eager developers are to adopt new skills**.
- **Expanding the "Programmable Surface Area"**
  - As AI increases programming efficiency, the **demand for software solutions will also expand**.
  - If programming opportunities **grow 20x**, we will still need **more developers**—even if each one is **10x more productive**.
- **Rising User Expectations**
  - Businesses that **only use AI to cut costs** will fall behind.
  - Companies that **invest in AI-driven innovation** will **lead the market** by offering superior services.

AI as a Catalyst for Ambition and Quality

AI Expands Developer Ambition

- **Simon Willison**, a veteran software developer, emphasizes that AI allows him to **be more ambitious** with his projects.
- AI doesn't just make coding easier—it **enables developers to tackle more complex problems**.

Parallel to the Film Industry's Evolution

- **Advances in computing power didn't just speed up animation; they improved quality.**
- **Example:**
  - A **single frame** of a modern Marvel movie may take as long to render as **an entire Pixar film** from the past.
  - Extra computational power led to better **fur, water, lighting, and higher resolutions** rather than just cheaper production.

The Trade-Off Between Faster & Higher Quality

- Some industries benefit from **speed and accessibility** (e.g., user-generated video content).
- However, **quality-driven markets will always exist**—AI will enable both **mass production** and **high-end innovation**.

The Democratization of Software Development

- **Tens of millions of amateur AI-assisted programmers** will experiment with tools like **Replit, Devin, Salesforce, Palantir, and Sierra**.
- Many will stumble upon **high-impact use cases** that could reach **millions of users**.
- Some will **become entrepreneurs**, while others will have their ideas **refined and scaled by professional developers**.

The Reinvention of Programming: AI and Workflow Integration

Workflow Still Matters

- **Sankar emphasizes** that programming isn't just about writing code—it's about **understanding workflows**.
- Developers must determine:
  - What can be done by **traditional software**
  - What can be done by **AI**
  - What must still be done by **humans**
  - How to **orchestrate these elements** effectively

The Importance of Adaptive Toolchains

- The most **valuable toolchains** will:
  - **Capture feedback**
  - **Learn edge cases**
  - **Optimize workflows quickly** for real-world use

AI's Role in Empowering Developers

- AI will **liberate developers**, allowing them to work **closer to the business domain** and maximize their impact.
- Top-tier **subject matter experts** will become **AI-assisted programmers**, leveraging AI to automate tasks and solve problems.

The Future of Work: AI-Assisted Programming

- **Programmers won't be replaced**—instead, every job role will require some level of **AI-assisted programming**.
- Those who fail to adopt AI tools will **fall behind**, regardless of their industry.

Not the End, But a New Beginning

- **Programming is not dying**—it is undergoing **its latest reinvention**, evolving into a higher-level, AI-augmented discipline.

Required Reading

Required Reading

- The end of programming as we know it
- Chapter 1,2, and 3 of “Prompt Engineering for LLMs”