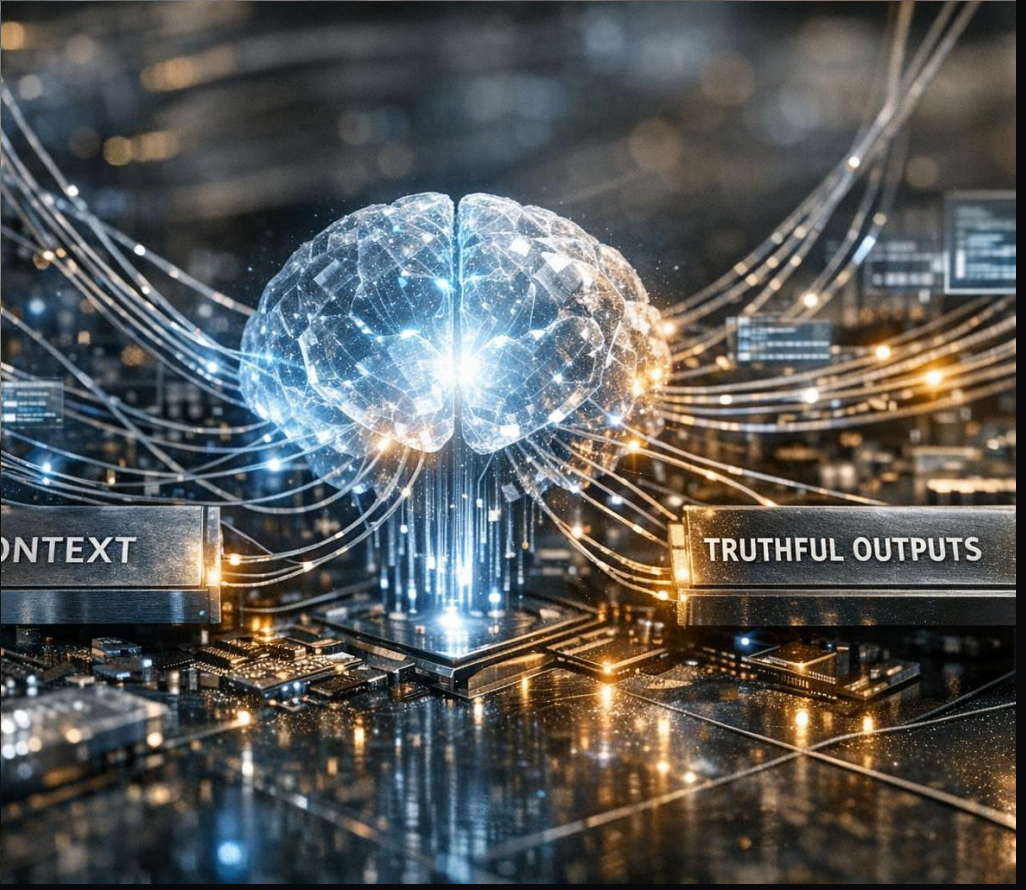
# Vibe Coding in Practice: Context Engineering and Driving AI Toward Truth

Practical approaches for enhancing AI accuracy and context understanding

# Today's Agenda

- Vibe Coding in Practice: Context Engineering and Driving AI Toward Truth

# Vibe Coding in Practice: Context Engineering and Driving AI Toward Truth




# Title Slide: Vibe Coding in Practice – Context Engineering and Driving AI Toward Truth

**Context Engineering Importance**

Context engineering improves LLM accuracy by aligning outputs closer to truth and reducing hallucinations.

**Overcoming Model Limitations**

Practical workflows and engineering mindsets help address inherent model limitations and improve reliability.

**AI-Driven Decision Making**

Context enhances AI-driven coding and decision making by providing relevant information for better outcomes.

# whoami: Jan Bronicki, Software Engineer, Microsoft

**Expertise in AI System Design**

Jan Bronicki specializes in designing AI systems, focusing on innovative prompting strategies and large language model applications.

**Bridging Software Engineering and AI**

He integrates traditional software engineering with generative AI to improve coding efficiency and AI correctness.

**Addressing AI Challenges**

His work addresses practical challenges in AI usage, ensuring correctness and reliability in software coding environments.


# Thesis: LLMs as Approximation Engines, Objective-Correctness Is Essential

**LLMs as Probability Approximators**

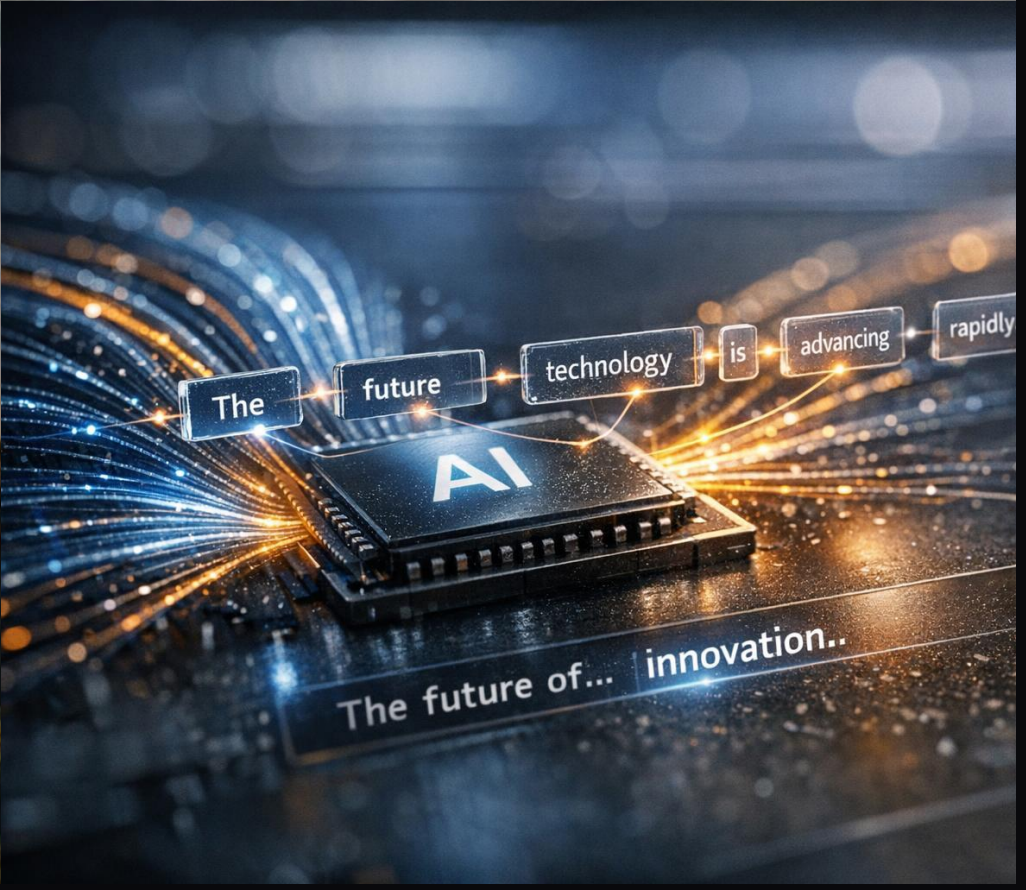
Large Language Models generate outputs by approximating token probabilities based on vast training datasets.

**Lack of Guaranteed Correctness**

LLM outputs maximize likelihood but do not ensure factual or objective correctness inherently.

**Importance of Objective-Correctness**

Incorporating external validation ensures AI outputs are reliable, truthful, and not just plausible.




# What an LLM Actually Is: Conditional Token Predictors and Training Data

### Conditional Token Prediction

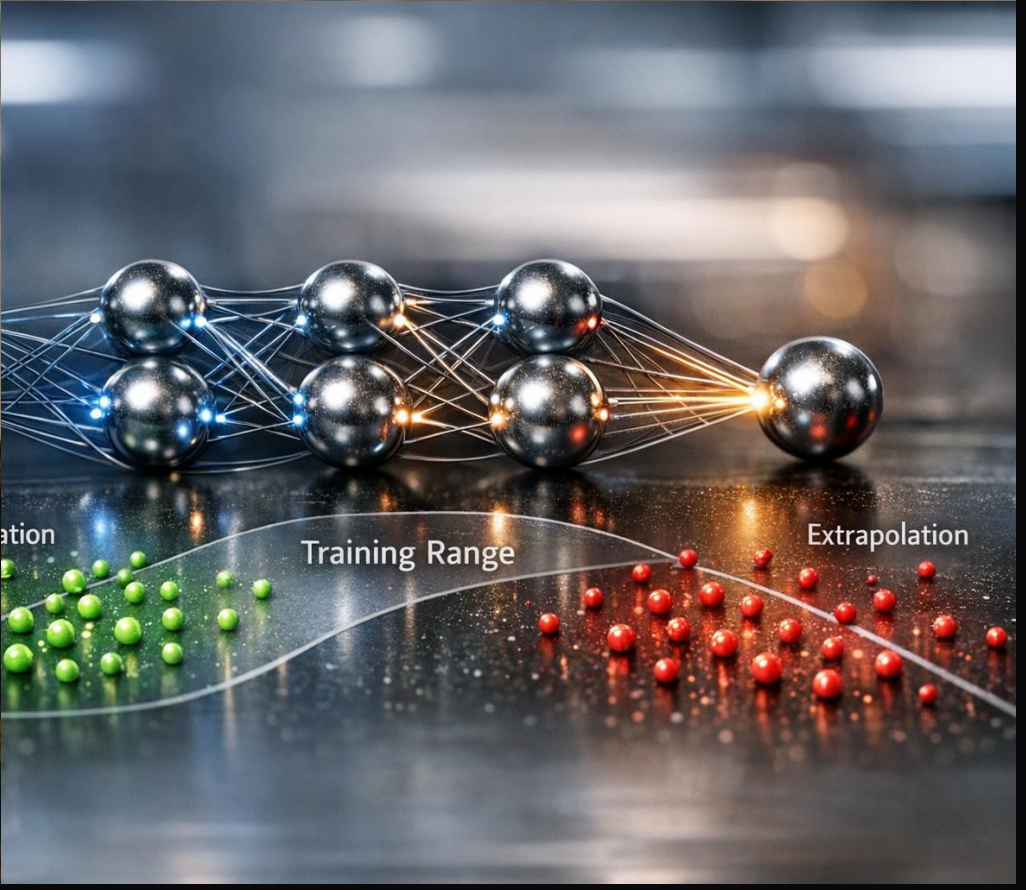
LLMs predict the next token based on previous tokens using learned probability distributions from training data.

### Training Data Diversity

Training data includes diverse sources, influencing the model's knowledge and behavior.

### Model Bias and Limitations

LLMs reflect biases and gaps in training data, leading to hallucinations or overconfidence in rare contexts.




# Interpolation vs Extrapolation: Strengths and Weaknesses of LLMs

**Interpolation Strengths**

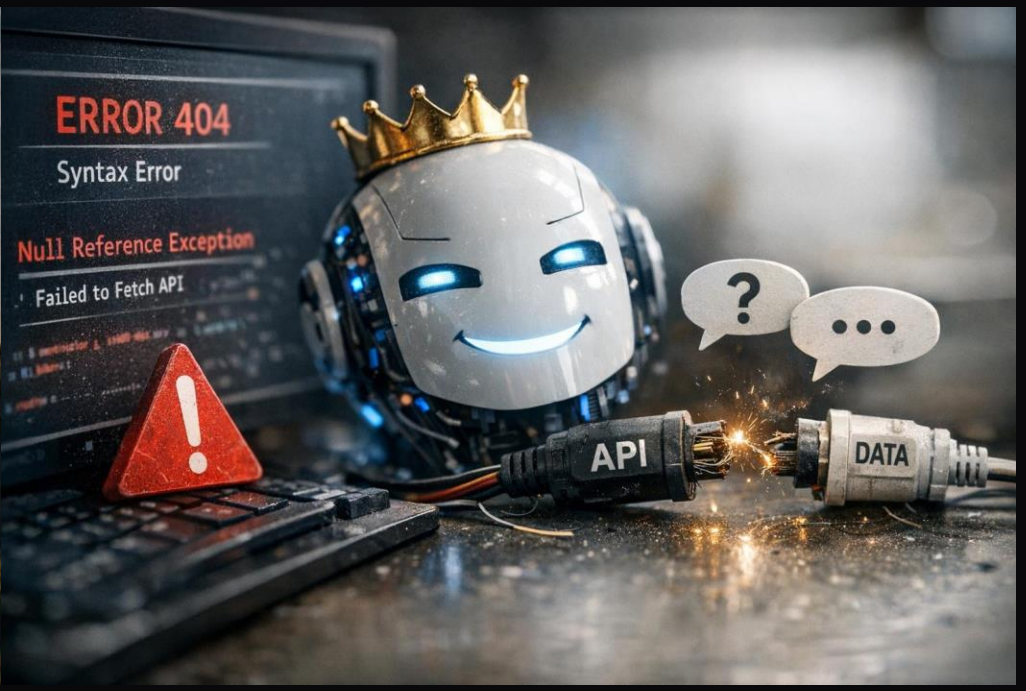
LLMs generate accurate text when input stays near known training data distributions, ensuring reliable outputs.

**Extrapolation Weaknesses**

LLMs struggle with inputs far from training data, causing errors in novel or creative scenarios.

**Prompt Framing Guidance**

Understanding LLM limits helps design prompts that keep outputs within safe interpolation zones.




# Why Failures Happen: Hallucinated APIs, Overconfidence, and Averaged Code

**Hallucinated APIs**

Large language models may generate nonexistent API functions, leading to misleading or incorrect outputs.

**Overconfidence in Outputs**

Models often present fabricated details confidently, making errors less obvious without careful review.

**Averaged Code Snippets**

Training data blending causes code outputs to merge multiple variants, creating subtle and hard-to-detect bugs.

**Need for Verification**

Raw model outputs require thorough verification to prevent errors from untrusted AI-generated content.


# Context Engineering Defined: Prompt, Retrieval, Constraints, Feedback

**Prompt Design**

Careful crafting of prompts guides AI to generate relevant and accurate responses aligned with user intent.

**Knowledge Retrieval**

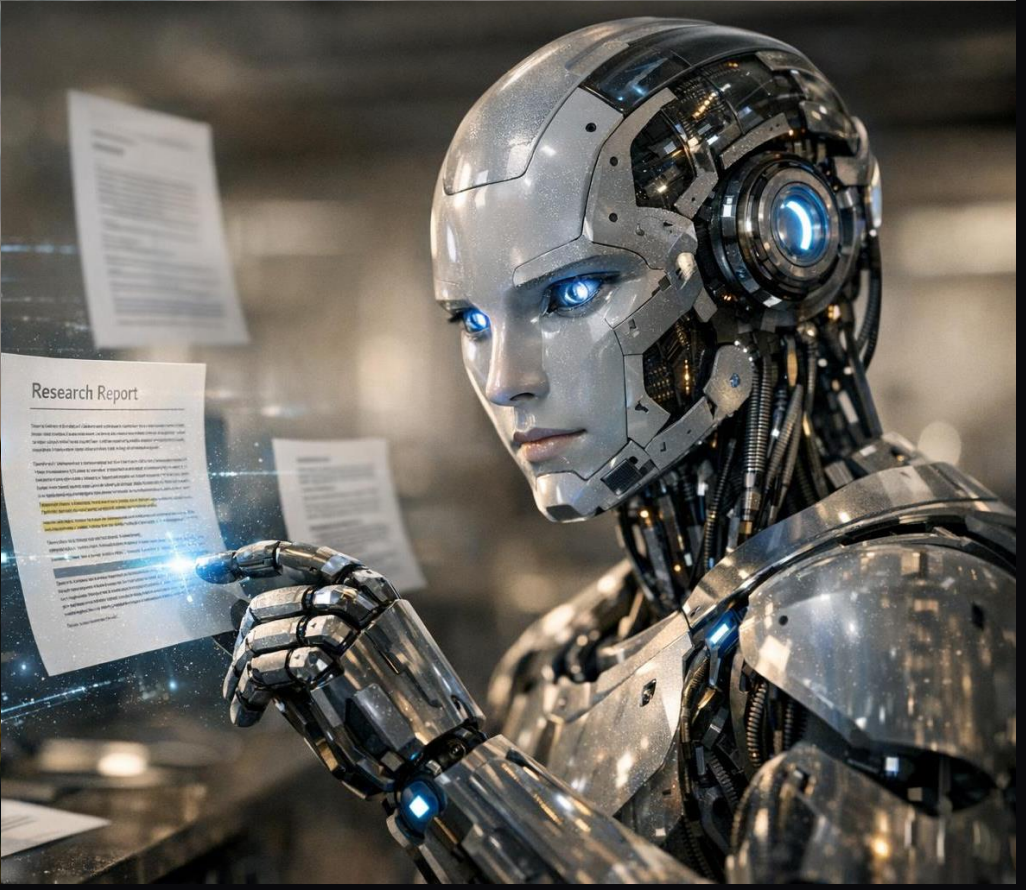
Incorporating external knowledge retrieval enhances AI output with relevant, up-to-date information.

**Enforcing Constraints**

Applying constraints limits AI scope to maintain focus and improve output quality.

**Iterative Feedback**

Feedback loops refine AI outputs continuously to better align with truth and user goals.




# RAG Demystified: Retrieval-Augmented Generation as Memory

**Integration of Document Retrieval**

RAG incorporates external document retrieval to supplement the model's internal knowledge during generation.

**Augmenting Latent Memory**

This method augments latent memory with current, domain-specific data to enhance response relevance.

**Improved Answer Accuracy**

Grounding outputs in verified sources improves answer accuracy beyond learned statistical patterns.

# Practical Example: Naive Prompt vs Context-Rich Prompt with Domain Model

### Naive Prompt Limitations

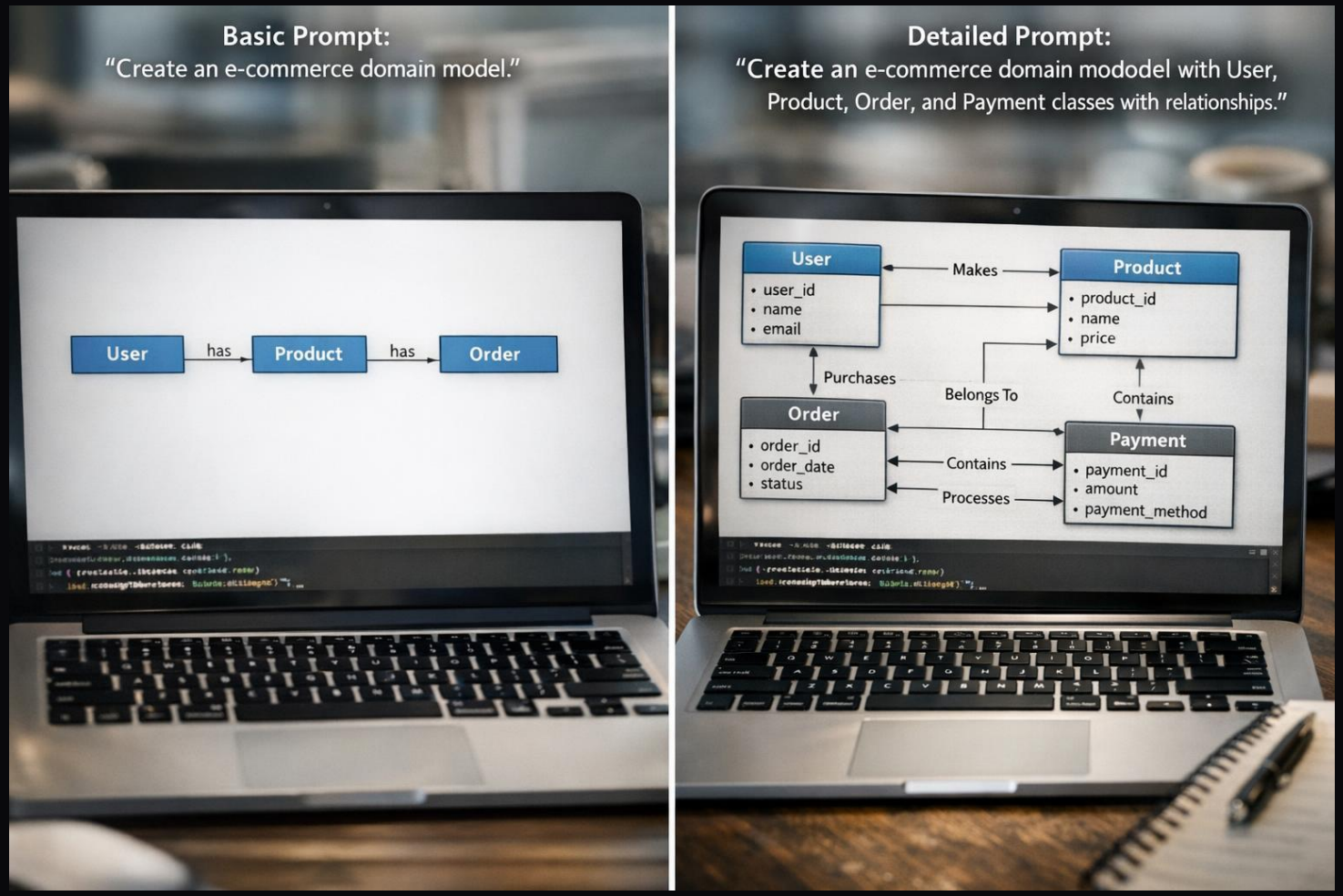
Naive prompts usually result in shallow or inaccurate responses lacking domain-specific details.

### Context-Rich Prompt Benefits

Context-rich prompts embed domain knowledge, improving output accuracy and relevance significantly.

### Impact of Context Engineering

Using context engineering techniques dramatically enhances code correctness and specificity in AI generation.




# Verification Loop: Testing, Typing, and Review for Correctness

## Automated Testing

Automated testing validates AI outputs systematically to detect errors and ensure reliability in software systems.

## Static Typing Checks

Static typing enforces data type correctness early to prevent runtime errors and improve code quality.

## Manual Code Reviews

Manual code reviews provide human insights to catch subtle mistakes and improve overall code correctness.

# Skill Shift: From Typing to Reviewing and System Design

**Shift from Manual Coding**

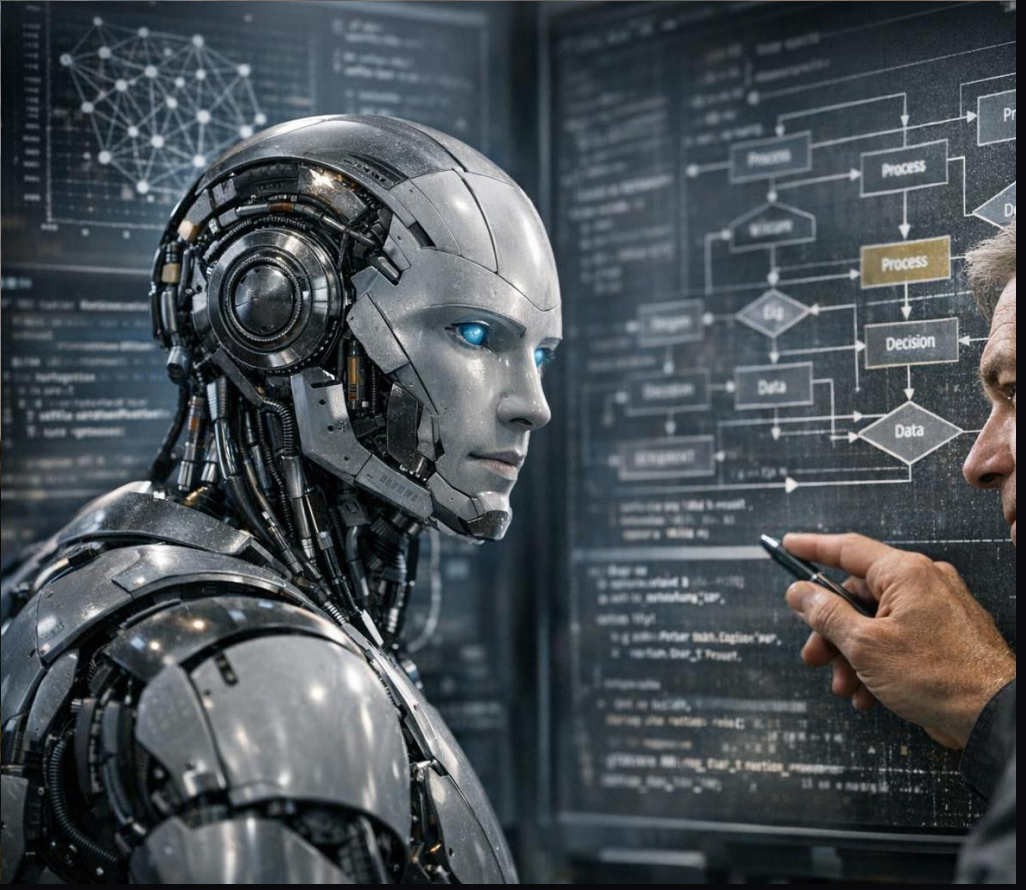
Engineers now spend less time typing code and more on reviewing and validating AI-generated code effectively.

**System Design Focus**

Designing AI-powered systems requires understanding architecture and safe integration of AI capabilities.

**New AI-Related Skills**

Proficiencies like prompt engineering and human-AI collaboration are essential in modern software development.




# Where It Breaks: Novel Algorithms and Deep Domain Logic Challenges

### Challenges with Novel Algorithms

AI faces difficulty implementing new algorithms due to limited training examples and lack of prior data.

### Deep Domain Logic Limitations

Specialized domain logic is hard for AI to master without extensive specific data and human insight.

### Importance of Human Expertise

Human experts are essential to design, specify, and validate innovative solutions beyond AI’s current capabilities.


# Final Message: LLMs Optimize Likelihood, Engineers Optimize Truth

**LLMs Optimize Likelihood**

Large language models predict the most probable next token instead of guaranteeing absolute truth in responses.

**Engineers Steer Truth**

Engineers play a critical role in guiding models through context engineering and validation to ensure truthful outputs.

**Symbiosis for Trustworthy AI**

The collaboration between LLMs and engineers defines the future of reliable and trustworthy AI-assisted software development.

# Driving AI Toward Truth through Context and Verification

## Enhancing LLMs with Context

Context engineering improves large language models by focusing on objective correctness over simple likelihood predictions.

## Techniques for Reliability

Techniques like retrieval-augmented generation and verification loops help improve AI output accuracy and truthfulness.

## Role of Engineers

Engineers play a vital role ensuring AI outputs are truthful and reliable despite models optimizing for prediction.