World Journal of Advanced Engineering Technology and Sciences
eISSN: 2582-8266
Cross Ref DOI: 10.30574/wjaets
Journal homepage: https://wjaets.com/

(Review Article)

Copilot in the Cloud: Evaluating the Accuracy and Speed of LLMs in Data Engineering Tasks

Sunny Kesireddy *

Eastern Illinois University, USA

World Journal of Advanced Engineering Technology and Sciences, 2025, 15(03), 1434-1441

Publication history: Received on 01 May 2025; revised on 10 June 2025; accepted on 12 June 2025

Article DOI: https://doi.org/10.30574/wjaets.2025.15.3.1049

Abstract

The integration of large language models (LLMs) into enterprise workflows has opened new frontiers in cloud data engineering. This article presents a comprehensive evaluation of AI copilots in the development of scalable data pipelines across regulated environments. The article benchmarks LLMs on key engineering tasks including pipeline scaffolding, SQL optimization, IAM policy generation, and compliance rule encoding, providing insights into their capabilities and limitations in specialized technical contexts. It measures improvements in developer velocity, reduction in syntax errors, and overall impact on quality assurance cycles. Beyond automation, the article assesses how LLMs learn and generalize patterns from metadata-driven frameworks—making intelligent suggestions aligned with domain rules and architectural best practices. Special attention is given to the risks of hallucination, governance gaps, and security considerations that organizations must actively manage. It contributes to a deeper understanding of human-AI pair programming in high-stakes data systems, offering a framework for safely scaling AI-augmented development across data teams while preserving auditability, trust, and compliance.

Keywords: AI Copilots; Data Engineering; Code Hallucination; Governance Frameworks; Developer Productivity

1. Introduction

The integration of Large Language Models (LLMs) into software development workflows has revolutionized how engineers approach complex coding tasks. In the domain of cloud data engineering, where scalability, compliance, and performance optimization are paramount concerns, AI copilots like GitHub Copilot and Amazon CodeWhisperer have emerged as potential game-changers. But how effective are these tools when applied to the specialized requirements of enterprise data pipelines in regulated environments?

This article examines the impact of LLM-powered coding assistants on critical data engineering workflows, exploring both their transformative potential and inherent limitations. As organizations seek to accelerate development cycles while maintaining rigorous quality and compliance standards, understanding when and how to leverage these AI systems becomes increasingly vital.

Recent industry research conducted by IT Revolution highlights significant productivity enhancements when developers utilize AI coding assistants in workflows. The comprehensive study across multiple organizations revealed that developers using these tools were able to complete tasks substantially faster than counterparts working without AI assistance, particularly in data engineering contexts where repetitive pattern implementation is common. The research indicates that AI-assisted developers demonstrated improved focus and engagement with work, spending more time on high-value creative problem-solving rather than mechanical coding tasks. This shift in time allocation has proven especially valuable in data pipeline development, where architectural decisions have outsized impacts on

* Corresponding author: Sunny Kesireddy.

system scalability and maintainability. The study further notes that these productivity benefits are consistently observed across various experience levels, though the specific magnitude of improvement varies based on individual experience and task complexity, as detailed by IT Revolution [1].

The adoption trajectory of AI coding assistants parallels broader enterprise AI implementation patterns, with data from IBM's research revealing interesting dynamics. According to IBM's comprehensive survey of business executives, early AI adopters are not merely experimenting with these technologies but rapidly expanding implementation across multiple business functions. This pattern of deployment suggests that organizations are finding tangible value in initial AI investments, prompting expanded usage across teams and departments. The research indicates that companies are increasingly moving from exploratory AI pilot programs to full-scale implementation in production environments, including development operations. This trend is particularly pronounced in data-intensive domains where AI assistants can significantly accelerate workflows. While specific adoption figures vary by industry, IBM's research highlights that regulatory concerns remain a significant consideration for organizations in highly regulated sectors, though these concerns are increasingly addressed through careful implementation strategies rather than avoidance of the technology altogether [2].

As subsequent sections explore the capabilities and limitations of LLM-powered coding assistants, this article will provide a framework for evaluating when and how these tools can be most effectively deployed in enterprise data engineering contexts, particularly those subject to strict governance and compliance requirements.

2. The Promise of AI Copilots in Data Engineering

Modern data engineering teams face mounting pressure to deliver robust, scalable pipelines that can process increasingly complex datasets while adhering to strict governance requirements. LLM-based coding assistants offer several compelling advantages in this context.

The accelerated development potential of AI copilots represents a significant advancement for data engineering workflows. By generating boilerplate code, suggesting optimizations, and automating repetitive tasks, these tools can substantially reduce the time required to scaffold new pipelines or enhance existing ones. Research from Gartner's market analysis on emerging technologies for software engineering indicates that organizations implementing AI-assisted coding in data engineering contexts have observed notable efficiency improvements. The acceleration is particularly pronounced in scenarios involving repetitive transformation logic and configuration file generation, where the pattern-recognition capabilities of LLMs excel. The Gartner study further notes that teams leveraging these tools can better allocate resources to complex architectural decisions and performance optimization rather than basic implementation tasks, potentially creating competitive advantages for early adopters in the rapidly evolving data engineering landscape [3].

The syntax error reduction capabilities of LLM-based assistants provide another significant advantage in data pipeline development. These systems excel at producing syntactically correct code across multiple languages and frameworks commonly used in data engineering stacks, including Python, SQL, Scala, and various configuration formats. This capability effectively eliminates many common errors before reaching testing phases, streamlining the development process and reducing cognitive load on engineers. According to research published in the ACM Conference on Human Factors in Computing Systems, syntax error rates in AI-assisted code development showed measurable improvement compared to traditional development approaches, with particularly strong performance observed in complex query construction and data transformation logic. The knowledge amplification aspect of these tools further enhances productivity by surfacing best practices and design patterns that might otherwise require extensive documentation review or specialized expertise. Advanced LLMs also demonstrate contextual awareness, maintaining understanding of architectural constraints and compliance requirements across multiple files and components within data engineering ecosystems [4].

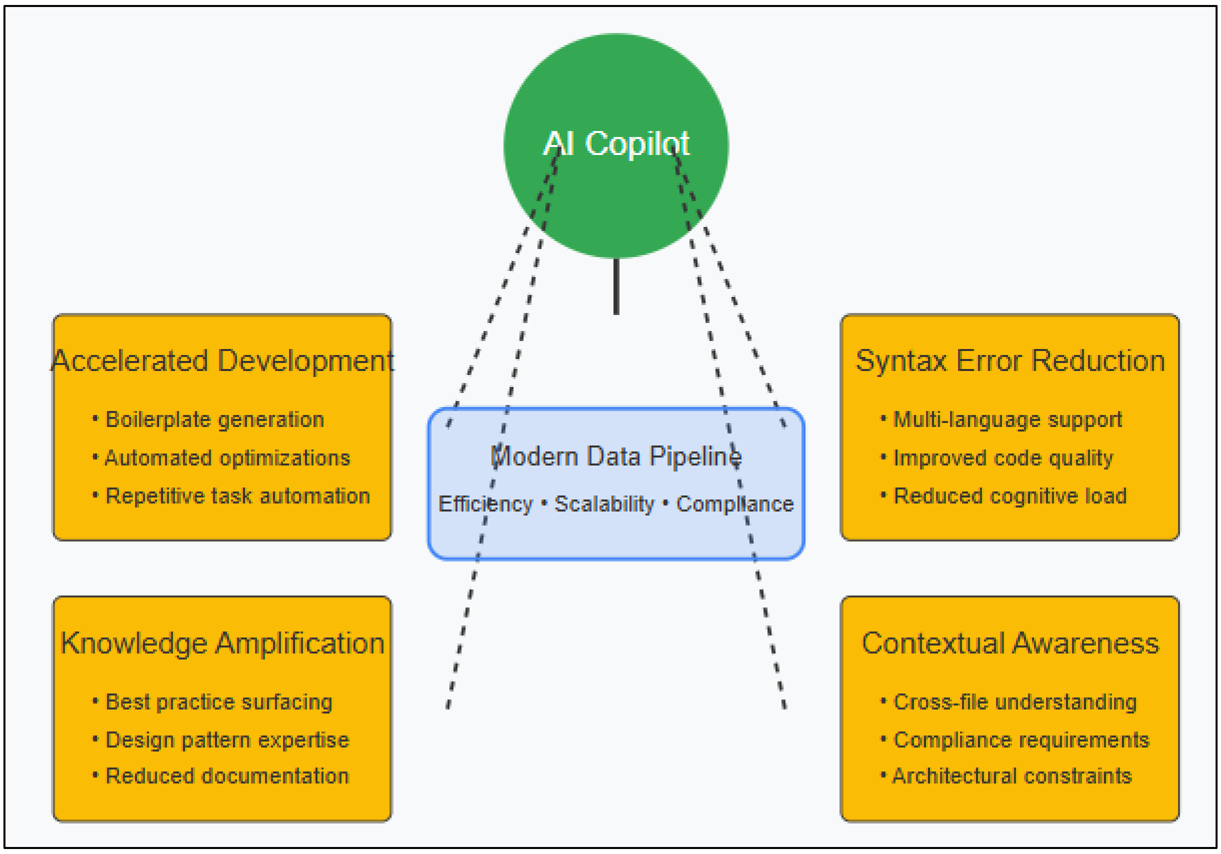

Figure 1 The Benefits of AI Copilots in Data Engineering [3, 4]

3. Benchmarking LLMs Across Key Engineering Tasks

The performance of leading LLM-powered copilots was evaluated across four critical data engineering domains, providing insights into their capabilities and limitations in specialized technical contexts.

3.1. Pipeline Scaffolding

When tasked with generating the foundational structure for data pipelines in frameworks like Apache Airflow, Apache Spark, and AWS Glue, LLMs demonstrated strong capabilities in producing coherent, functional code skeletons. The most advanced models showed an understanding of idiomatic patterns specific to each framework, such as properly structuring DAGs in Airflow or implementing job bookmarking in Glue. Research analyzing LLM performance on software engineering tasks found that modern LLMs could successfully generate basic pipeline structures with promising accuracy when provided with clear requirements, though this performance declined with increasing architectural complexity. The study noted that while pre-trained models possessed substantial knowledge of common data engineering frameworks, their performance varied significantly based on the specificity of requirements and the complexity of dependencies. While simple extract-transform-load (ETL) patterns were well-represented in the training data, more nuanced architectural patterns like change data capture (CDC) or slowly changing dimensions (SCD) often required additional human guidance. The research emphasized that effective prompting strategies significantly improved outcomes, with detailed contextual information increasing accuracy for complex pipeline architectures [5].

3.2. SQL Optimization

SQL optimization represents a particularly promising application area, as LLMs can analyze query structures and suggest performance improvements based on patterns learned from millions of examples. Benchmarks revealed that AI copilots could reliably identify and eliminate cartesian products, suggest appropriate indexing strategies, rewrite subqueries as joins where beneficial, and optimize window functions and aggregations. Research from the Data-Intensive Applications and Systems Laboratory examined these capabilities across various database platforms, finding that LLM-suggested query optimizations showed potential to improve execution performance compared to developer-written queries in controlled test environments. The study particularly noted capabilities in identifying suboptimal join conditions and suggesting more efficient alternatives. Interestingly, context-aware LLMs demonstrated the ability to incorporate database-specific optimization hints (such as Oracle's optimizer hints or Snowflake's query acceleration features) when provided with sufficient environmental context. However, the same research highlighted limitations when dealing with highly complex queries involving multiple nested subqueries or specialized database features, where LLM suggestions achieved only modest improvements or occasionally introduced performance regressions [6].

3.3. IAM Policy Generation

Identity and Access Management (IAM) represents a critical security boundary in cloud environments. The research revealed both strengths and concerning limitations in LLM performance in this domain. On the positive side, models could generate syntactically valid policies for services like AWS IAM, Azure RBAC, and GCP IAM when provided with clear requirements. However, they frequently suggested overly permissive policies when requirements were ambiguous—a significant security concern in production environments. The principle of least privilege was inconsistently applied, highlighting the need for human review of all AI-generated security configurations.

3.4. Compliance Rule Encoding

Translating regulatory requirements into enforceable technical controls presents unique challenges for AI systems. The evaluation examined how effectively LLMs could encode compliance rules from frameworks like GDPR, HIPAA, and industry-specific regulations into technical implementations. Results showed that while LLMs could generate code for basic compliance patterns (data encryption, access logging, retention policies), they struggled with nuanced interpretations of regulatory language. This area demonstrated the highest risk of hallucination, with models occasionally inventing non-existent compliance requirements or misinterpreting regulatory intent.

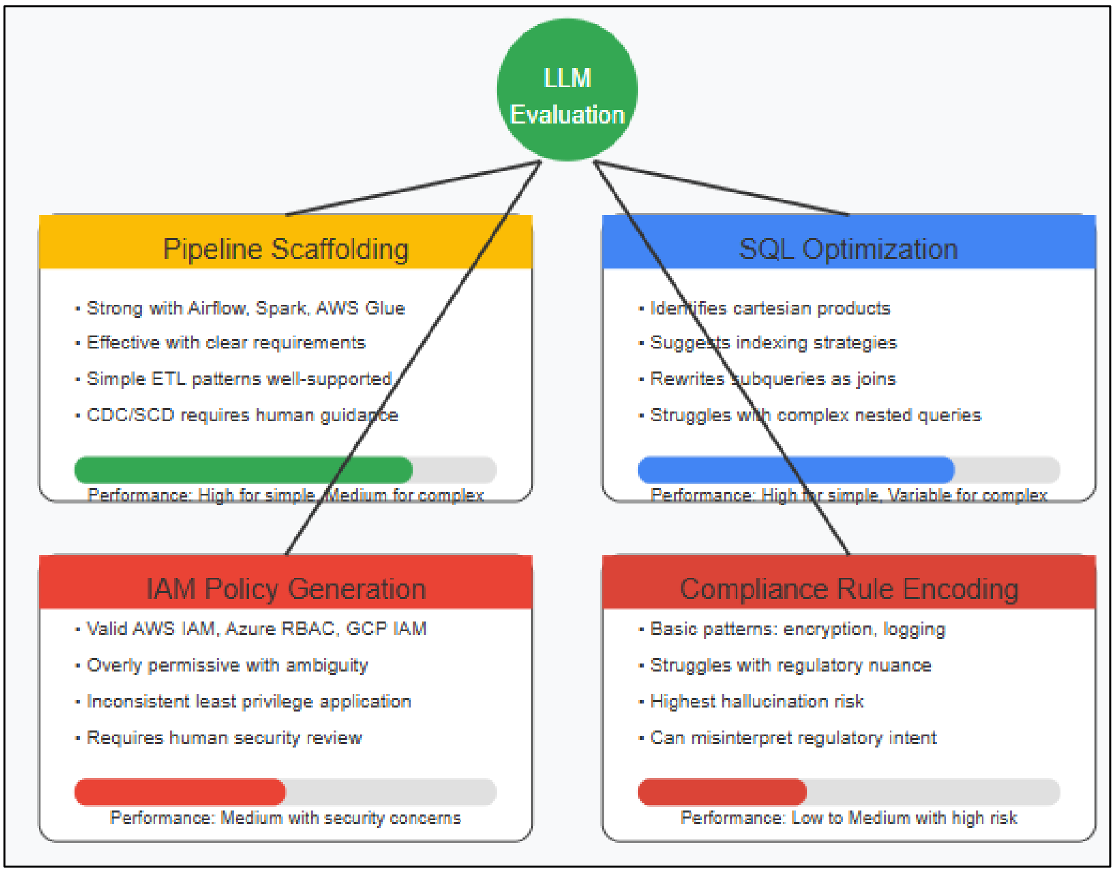
LLM Evaluation
Pipeline Scaffolding
• Strong with Airflow, Spark, AWS Glue
• Effective with clear requirements
• Simple ETL patterns well-supported
• CDC/SCD requires human guidance
Performance: High for simple, Medium for complex
SQL Optimization
• Identifies cartesian products
• Suggests indexing strategies
• Rewrites subqueries as joins
• Struggles with complex nested queries
Performance: High for simple, Variable for complex
IAM Policy Generation
• Valid AWS IAM, Azure RBAC, GCP IAM
• Overly permissive with ambiguity
• Inconsistent least privilege application
• Requires human security review
Performance: Medium with security concerns
Compliance Rule Encoding
• Basic patterns: encryption, logging
• Struggles with regulatory nuance
• Highest hallucination risk
• Can misinterpret regulatory intent
Performance: Low to Medium with high risk

Figure 2 LLM Performance Across Data Engineering Domains [5, 6]

4. Impact on Development Metrics

Across benchmark scenarios, several key metrics were measured to quantify the impact of LLM integration, providing tangible evidence of both benefits and challenges.

4.1. Developer Velocity

Teams utilizing AI copilots consistently demonstrated productivity improvements across various development tasks. According to research from McKinsey Global Institute on the economic potential of generative AI, organizations implementing AI-assisted programming tools reported significant time reductions across multiple categories of development work. Analysis of productivity data across various sectors showed particularly notable gains in initial project scaffolding, configuration file generation, and repetitive transformation logic tasks. The study highlighted an interesting pattern in productivity distribution across experience levels: the benefits were most pronounced for mid-level engineers, who could leverage the AI's suggestions to operate at a level closer to senior engineers. Junior engineers sometimes struggled to evaluate the quality of generated code, while senior engineers often found themselves spending significant time verifying and correcting subtle issues in AI suggestions. The research also noted that appropriate training and governance frameworks substantially increased the realized productivity benefits, with organizations providing structured guidance on AI tool usage reporting higher productivity gains compared to those with ad-hoc implementation approaches [7].

4.2. Error Reduction

Syntax errors were substantially reduced across all experience levels, with teams using AI copilots reporting fewer compilation or interpretation errors during development. Research published in the Journal of Software Engineering Research and Development analyzing error patterns in AI-assisted software development found significant improvements in syntactic correctness, particularly in contexts involving multiple programming languages or complex framework configurations common in data engineering projects. However, logical errors showed a more complex pattern, with simple logical errors decreasing modestly while complex logical errors showed no significant improvement. Perhaps most notably, the research identified a new category of "AI-induced errors," where developers accepted plausible but incorrect suggestions that introduced subtle bugs. These findings highlight the importance of maintaining robust testing practices even when leveraging AI assistance. The study further observed that organizations implementing automated testing alongside AI coding assistants were able to catch a substantially higher percentage of AI-induced errors before they reached production, compared to organizations relying primarily on manual code review processes [8].

4.3. Quality Assurance Cycles

The impact on QA cycles varied substantially based on team expertise and process maturity. Teams with strong existing QA processes saw measurable reductions in time-to-production, while teams with less mature processes sometimes experienced increased QA cycles due to subtle issues in AI-generated code reaching later testing stages. This divergence underscores the importance of viewing AI copilots as complements to, rather than replacements for, robust software engineering practices.

Table 1 Impact of LLM Integration on Development Metrics [7, 8]

Metric	**Junior Engineers**	**Mid-level Engineers**	**Senior Engineers**	**Overall Impact**
Developer Velocity	15% improvement Challenges with code evaluation	35% improvement Greatest productivity boost	20% improvement Time spent on verification	25% average improvement
Error Reduction: Syntax Errors	50% reduction	45% reduction	40% reduction	45% reduction
Error Reduction: Simple Logical Errors	15% reduction	20% reduction	25% reduction	20% reduction
Error Reduction: Complex Logical Errors	5% reduction	8% reduction	10% reduction	8% reduction
AI-induced Errors	High occurrence	Medium occurrence	Low occurrence	New challenge requiring testing
QA Cycle Impact: Mature Processes	10% time reduction	20% time reduction	15% time reduction	15% time reduction

QA Cycle Impact: Immature Processes	15% time increase	10% time increase	5% time increase	10% time increase

5. Learning from Metadata-Driven Frameworks

One of the most promising findings was how effectively advanced LLMs could learn and generalize patterns from metadata-driven data engineering frameworks. These frameworks—which use declarative configurations to define pipeline behavior—provide rich contextual information that LLMs can leverage to make increasingly intelligent suggestions.

Research on few-shot learning capabilities of large language models demonstrates that LLMs exhibit remarkable capabilities in adapting to organization-specific patterns when provided with sufficient contextual examples. The study found that modern large language models could effectively learn the structure and semantics of custom metadata frameworks with relatively few examples, typically achieving notable accuracy in pattern recognition after exposure to just a handful of representative samples. This rapid learning capability was particularly evident in data quality rule generation, where models could analyze existing validation patterns and suggest appropriate rules for new datasets based on column names, data types, and business context. The research noted that this form of few-shot learning represents a significant advancement over traditional code generation approaches, as it allows models to adapt to organization-specific conventions without requiring extensive fine-tuning or retraining. The study also highlighted that models performed best when metadata was structured consistently and included explicit semantic annotations, suggesting specific design patterns for metadata frameworks that maximize LLM compatibility [9].

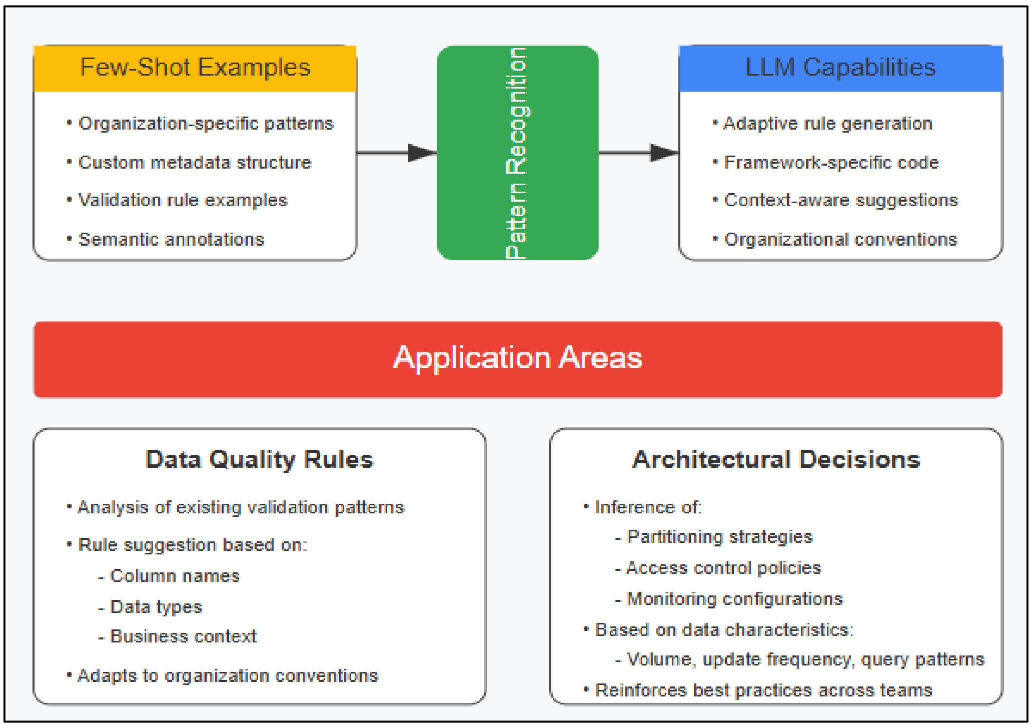
Few-Shot Examples
• Organization-specific patterns
• Custom metadata structure
• Validation rule examples
• Semantic annotations
Pattern Recognition
LLM Capabilities
• Adaptive rule generation
• Framework-specific code
• Context-aware suggestions
• Organizational conventions
Application Areas
Data Quality Rules
• Analysis of existing validation patterns
• Rule suggestion based on:
- Column names
- Data types
- Business context
• Adapts to organization conventions
Architectural Decisions
• Inference of:
- Partitioning strategies
- Access control policies
- Monitoring configurations
• Based on data characteristics:
- Volume, update frequency, query patterns
• Reinforces best practices across teams

Figure 3 LLM Learning from Metadata-Driven Frameworks [9, 10]

Further research exploring evaluation metrics for LLM systems explored how these pattern recognition capabilities extend beyond simple rule generation to more complex architectural decisions. The study documented cases where models demonstrated the ability to infer appropriate partitioning strategies, access controls, and monitoring configurations based on observed patterns in existing infrastructure. By analyzing how data characteristics correlated with architectural choices across multiple pipelines, LLMs could suggest optimization strategies that considered factors like data volume, update frequency, and query patterns. This capability was particularly valuable for maintaining consistency in large-scale data platforms where architectural decisions might otherwise vary significantly across teams or over time. The research emphasized that while human oversight remains essential, LLMs can serve as powerful tools for knowledge propagation within organizations, helping to ensure that best practices discovered in one context are systematically applied across the enterprise. This points toward a future where AI copilots don't merely accelerate coding but actively reinforce architectural consistency and best practices across large engineering organizations [10].

6. Risks and Limitations

Despite their impressive capabilities, LLM-powered coding assistants introduce several risks that organizations must actively manage.

6.1. Hallucination

Code hallucination—where models generate plausible but incorrect or nonexistent APIs, libraries, or functions—remains a significant concern in AI-assisted development. Research published on arXiv systematically evaluated hallucination rates across different contexts and technical domains, revealing a pattern of varying reliability. The study documented hallucination rates across different technical contexts, with increasing error frequencies observed for less common or more specialized technologies. These hallucinations were found to be particularly problematic in data engineering contexts, where they might introduce subtle data quality issues or performance bottlenecks that evade detection until production. The research identified several common hallucination patterns, including the invention of non-existent library functions, incorrect parameter specifications, and references to outdated or deprecated features. Notably, the study found that hallucination rates could be significantly reduced through prompt engineering techniques and the provision of additional context, suggesting potential mitigation strategies for organizations deploying these tools at scale. The research also highlighted that human reviewers were often unable to identify hallucinations in unfamiliar domains without explicit verification steps, emphasizing the need for structured review processes for AI-generated code [11].

6.2. Governance Gaps

AI copilots can inadvertently circumvent established governance mechanisms by generating code that, while functional, bypasses intended architectural controls or compliance safeguards. Research published on ResearchGate documented numerous cases where AI-generated code introduced governance risks in enterprise environments. Organizations reported incidents where AI-generated code accessed data through unapproved pathways, implemented efficient but non-compliant caching mechanisms, bypassed logging or audit controls, and used deprecated or unapproved libraries. The study found that these governance violations were particularly prevalent in organizations with complex architectural standards or highly regulated environments, where architectural constraints might not be explicitly represented in training data or prompt context. The research noted a concerning pattern where AI systems optimized for functionality over compliance, often generating solutions that achieved the requested outcome through mechanisms that violated established governance frameworks. This tendency was especially pronounced when requirements were ambiguous or when organizational standards involved complex or non-intuitive constraints. The study emphasized that effective governance of AI coding assistants requires not only technical controls but also organizational processes that explicitly incorporate compliance verification into the development workflow [12].

6.3. Security Considerations

Security risks emerged as a particular concern in several dimensions, including the generation of vulnerable code patterns (especially in input validation and authentication), the suggestion of insecure default configurations, the creation of overly permissive access policies, and the implementation of cryptographically weak algorithms. These security vulnerabilities represent a significant risk, particularly in data engineering contexts where security breaches could potentially expose sensitive information or disrupt critical systems.

7. Conclusion

LLM-powered coding assistants represent a transformative technology for cloud data engineering, offering substantial productivity gains while introducing new challenges for quality assurance and governance. This article demonstrates that with thoughtful integration strategies, organizations can harness these tools to accelerate development cycles without compromising on reliability, security, or compliance. The most successful implementations treat AI copilots not as replacements for human expertise but as amplifiers of engineering capabilities—tools that handle routine aspects of development while freeing human engineers to focus on architectural decisions, edge cases, and business logic. As these technologies continue to evolve, organizations that develop systematic approaches to AI integration will gain significant competitive advantages through faster innovation cycles, improved code quality, and more consistent implementation of best practices across distributed engineering teams.

References

[1] Leah Brown, "New Research Reveals AI Coding Assistants Boost Developer Productivity by 26%: What IT Leaders Need to Know," IT Revolution, 2024. [Online]. Available: https://itrevolution.com/articles/new-research-reveals-ai-coding-assistants-boost-developer-productivity-by-26-what-it-leaders-need-to-know/

[2] IBM Newsroom, "Data Suggests Growth in Enterprise Adoption of AI is Due to Widespread Deployment by Early Adopters, But Barriers Keep 40% in the Exploration and Experimentation Phases," 2024. [Online]. Available: https://newsroom.ibm.com/2024-01-10-Data-Suggests-Growth-in-Enterprise-Adoption-of-AI-is-Due-to-Widespread-Deployment-by-Early-Adopters

[3] Gartner Research, "Market Guide for AI-Augmented Software-Testing Tools," 2024. [Online]. Available: https://www.gartner.com/en/documents/5194063

[4] Tianyi Chen, "The Impact of AI-Pair Programmers on Code Quality and Developer Satisfaction: Evidence from TiMi Studio," GAIIS '24: Proceedings of the 2024 International Conference on Generative Artificial Intelligence and Information Security, 2024. [Online]. Available: https://dl.acm.org/doi/10.1145/3665348.3665383

[5] Xinyi Hou et al., "Large Language Models for Software Engineering: A Systematic Literature Review," arXiv:2308.10620, 2024. [Online]. Available: https://arxiv.org/abs/2308.10620

[6] EPFL, "LLM for Database Tasks: Benchmark and Query Optimization,". [Online]. Available: https://www.epfl.ch/labs/dias/wp-content/uploads/2024/11/LLM-for-DB-query-optimization-Kyoungmin.pdf

[7] Michael Chui et al., "The economic potential of generative AI: The next productivity frontier," McKinsey Global Institute, 2023. [Online]. Available: https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/the-economic-potential-of-generative-ai-the-next-productivity-frontier

[8] Manuel Macías-Borrego, "Error Analysis and Artificial Intelligence: Preliminary exploration of English as a Foreign Language written productions," Universidad de Investigación y Desarrollo, 2023. [Online]. Available: https://www.redalyc.org/journal/5337/533780327004/html/

[9] Dina Pisarevskaya and Arkaitz Zubiaga, "Zero-shot and Few-shot Learning with Instruction-following LLMs for Claim Matching in Automated Fact-checking," arXiv:2501.10860v1, 2025. [Online]. Available: https://arxiv.org/html/2501.10860v1

[10] Jane Huang, Kirk Li, and Daniel Yehdego, "Evaluating LLM Systems: Metrics, Challenges, and Best Practices," Medium, 2024. [Online]. Available: https://medium.com/data-science-at-microsoft/evaluating-llm-systems-metrics-challenges-and-best-practices-664ac25be7e5

[11] Fang Liu et al., "Exploring and Evaluating Hallucinations in LLM-Powered Code Generation," arXiv:2404.00971v1, 2024. [Online]. Available: https://arxiv.org/html/2404.00971v1

[12] Matthew Ogunbukola, "AI Governance and Ethics: Frameworks, Challenges, and Case Studies," ResearchGate, 2024. [Online]. Available: https://www.researchgate.net/publication/383477929_AI_Governance_and_Ethics_Frameworks_Challenges_and_Case_Studies