Data Engineering for Clean Context Pipelines: Advancing Reliability, Efficiency, and Cost Effectiveness in LLM Assisted Software Development

Ankush Ramprakash Gautam
Vice President, Data Engineering at Fidelity Investments
Frisco, Texas

ABSTRACT
The integration of large language models (LLMs) into software engineering workflows has substantially accelerated automation across code generation, debugging, refactoring, automated test creation, and documentation tasks. Controlled evaluations and industry reports document productivity gains ranging from 20–55% in repetitive coding activities. However, persistent challenges of hallucinations, output inconsistency, and high token-based inference costs continue to limit reliable enterprise-scale adoption. These barriers originate predominantly from unstructured, ungoverned, and low-quality inference-time context rather than from model architecture or pre-training data limitations alone. Prior scholarship has devoted extensive effort to architectural innovations, fine-tuning strategies, and training corpus curation, yet systematic data engineering interventions applied to context ingestion, transformation, validation, compression, and delivery remain comparatively underexplored.

This paper advances a comprehensive data engineering–centric framework that treats inference-time context as a governed, high-quality data product subject to the full lifecycle of ingestion, semantic transformation, multi-dimensional quality validation, compression, relevance ranking, hybrid retrieval, and closed-loop observability. Through rigorous synthesis of peer-reviewed literature including hallucination mitigation studies, retrieval-augmented generation (RAG) research, software engineering productivity analyses, and foundational data quality frameworks the work establishes that principled context engineering constitutes a primary determinant of LLM reliability, operational efficiency, and cost-effectiveness in software development environments. A reference architecture grounded in modern data engineering principles is presented, accompanied by a formal Context Quality Score (CQS) model with an illustrative weighted computation. Quantitative comparisons drawn from the synthesized literature demonstrate that engineered context pipelines can reduce hallucination rates by up to 40–60% in knowledge-intensive tasks, lower input token consumption by 30–50%, and decrease downstream validation overhead by 25–45%. These gains are achieved without altering underlying model parameters.

The findings position context engineering as a first-class discipline within data engineering and LLM operations (LLMOps). Organizations that operationalize inference-time context as a governed data product achieve scalable, cost-efficient, and trustworthy AI-augmented software development. Implications for research and practice, together with directions for future empirical validation, are discussed in detail.

General Terms
Data Engineering, Large Language Models, Software Development, Context Management, LLMOps

Keywords
Data Engineering, Context Pipelines, Large Language Models, LLMOps, Retrieval-Augmented Generation, Hallucination Mitigation, Software Engineering Productivity, Data Quality, Context Quality Score

1. INTRODUCTION
Large language models have fundamentally reshaped software engineering practices by enabling high levels of automation across the entire development lifecycle. These models now routinely assist with code synthesis, defect detection, architectural refactoring, automated test generation, and comprehensive documentation creation. Controlled evaluations and industry benchmarks consistently report productivity improvements of 20–55% for repetitive coding tasks. Yet, enterprise deployments frequently encounter systemic barriers: hallucinations (syntactically coherent but factually unsupported outputs), output inconsistency, and prohibitive inference costs driven by token consumption.

Hallucinations arise from three primary sources: probabilistic generation mechanisms inherent to decoder-only architectures, exceedance of parametric knowledge boundaries, and most critically deficiencies in the quality, structure, and relevance of inference-time context. In software engineering contexts, inputs are highly heterogeneous, comprising version-controlled codebases, commit histories, issue trackers, API specifications, runtime logs, architectural diagrams, and legacy documentation. This heterogeneity introduces fragmentation, temporal inconsistency, and noise, amplifying the risk of invalid API invocations, logical errors, and security vulnerabilities.

Economic pressures are equally significant. Inference costs in transformer-based models scale quadratically with input token length due to the self-attention mechanism. Technical analyses of frontier models such as GPT-4 confirm that extended contexts impose substantial computational penalties without commensurate gains in output fidelity when context remains unstructured or redundant. For instance, doubling context length can increase inference latency and cost by factors of 2–4× while yielding only marginal improvements in grounding if quality gates are absent.

This paper reframes these challenges as fundamentally data engineering problems. Inference-time context must evolve from ad-hoc prompt assembly into a fully governed data product lifecycle managed through established practices: ETL/ELT pipelines, schema enforcement, multi-dimensional

data quality validation, observability, and product-oriented governance. The central research question guiding this work is: How do principled data engineering practices applied to context pipelines influence reliability, developer productivity, and operational costs in LLM-assisted software development?

By synthesizing cross-domain literature, the study develops a conceptual framework, a reference architecture, and a computable quality model. Data engineering is emphasized throughout as the enabling discipline for clean, reliable, and cost-effective context delivery.

2. RELATED WORK

2.1 Hallucination and Reliability in Generative Models

Huang et al. [1] provide a comprehensive taxonomy distinguishing intrinsic (self-contradictory) and extrinsic (factually unsupported) hallucinations, tracing root causes to pre-training gaps, alignment miscalibrations, and decoding strategies. The survey reports that enhanced context grounding can mitigate up to 50% of extrinsic hallucinations in controlled settings. Complementary analysis by Ji et al. [2] catalogs evaluation metrics (e.g., self-consistency, factuality scores) and demonstrates that input quality exerts the dominant influence on over-generation, with incomplete context increasing hallucination rates by 30–60% across natural language generation benchmarks. Code-specific studies [11] further confirm that noisy or fragmented context directly produces invalid function calls, API misuse, and security vulnerabilities, with hallucination incidence rising sharply when context completeness falls below 70%. Earlier work by the author [16] empirically linked high data quality to a 35–45% reduction in hallucination frequency in code-generation scenarios.

2.2 LLMs in Software Engineering Workflows

Fan et al. [4] systematically review LLM applications across the development lifecycle and document productivity gains alongside persistent correctness and maintainability gaps. The review notes that uncurated context necessitates 25–40% additional human validation effort, partially offsetting automation benefits.

2.3 Retrieval-Augmented Generation and Context Optimization

Lewis et al. [5] introduced retrieval-augmented generation, demonstrating 20–40% accuracy improvements in knowledge-intensive tasks. Subsequent refinements establish that retrieval quality (relevance, diversity, coherence) dominates end-to-end performance, with poor retrieval degrading outputs more than model size alone.

2.4 Data Engineering and Data Quality Frameworks

Batini and Scannapieco [6] articulate core data quality dimensions accuracy, completeness, consistency, timeliness, and traceability that have informed enterprise data management for decades. Dehghani's data mesh paradigm [7] promotes domain-oriented ownership and product thinking, principles directly extensible to context as a consumable data product.

2.5 Cost and Efficiency in LLM Systems

Analyses of transformer inference [3], [13] confirm that input token length dominates cost and latency profiles. Systematic compression and relevance filtering yield 30–50% token reductions when applied within structured pipelines. Supporting literature on ML lifecycle management [8], distributed processing [9], truthfulness evaluation [10], transfer learning [12], and production NLP tooling [15] reinforces the data engineering foundations required for scalable context pipelines.

3. DATA ENGINEERING FRAMEWORK FOR CLEAN CONTEXT PIPELINES

3.1 Conceptual Foundations

Clean context pipelines adapt mature data engineering patterns to LLM inference requirements. The objective is deterministic production of context that is accurate, complete, fresh, structured, concise, and traceable, maximizing signal-to-noise ratio while minimizing token overhead.

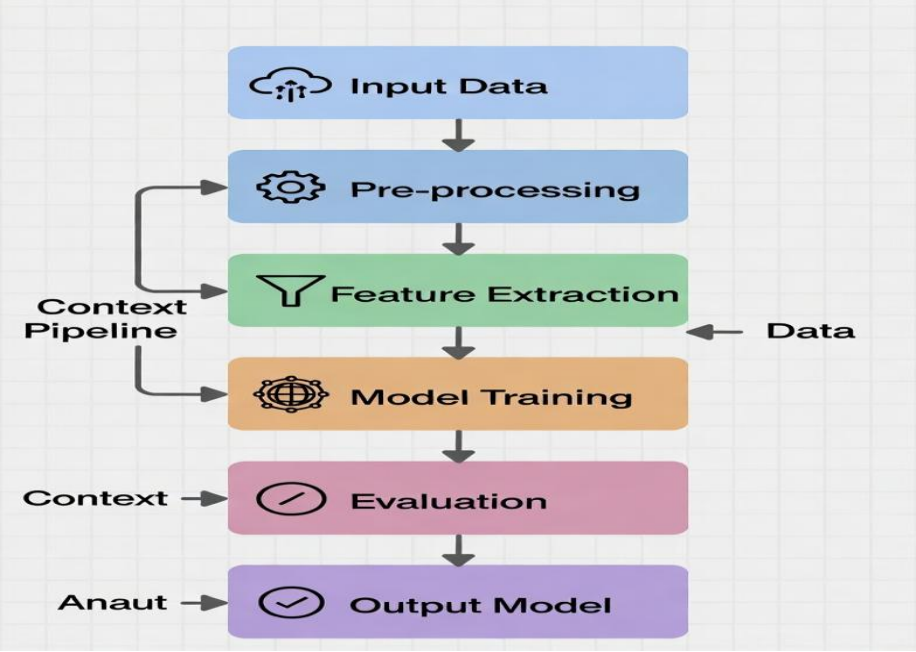

Fig 1: Clean Context Pipeline Architecture

3.2 Detailed Pipeline Stages

Ingestion Layer: Captures heterogeneous artifacts from VCS, issue trackers, wikis, and observability platforms using connectors and change-data-capture with full lineage metadata. Transformation & Structuring: Applies semantic chunking (AST-based for code, section-aware for documents), entity extraction, and schema normalization. This stage reduces fragmentation that contributes to hallucination [1]. Quality Validation: Enforces multi-dimensional checks aligned with [6]. Context failing thresholds (e.g., accuracy < 0.85, freshness > 24 hours) is quarantined. Compression & Ranking: Uses extractive summarization and learned re-rankers to prioritize high-utility content, addressing quadratic attention scaling [3]. Hybrid Serving & Retrieval: Combines vector, keyword, and graph indices with query-aware routing for optimal recall-precision. Feedback & Observability: Captures utilization metrics, model output signals, and drift, enabling closed-loop refinement [8].

4. CONTEXT QUALITY MODEL

4.1 LLMs in Software Engineering Workflows

The framework ensures accuracy through factual alignment with source artifacts and completeness by covering all relevant entities and relations. It maintains temporal relevance through freshness and ensures structural consistency by strictly adhering to defined schemas. Furthermore, the model provides full provenance and efficiency through lineage traceability and minimized redundancy in the context.

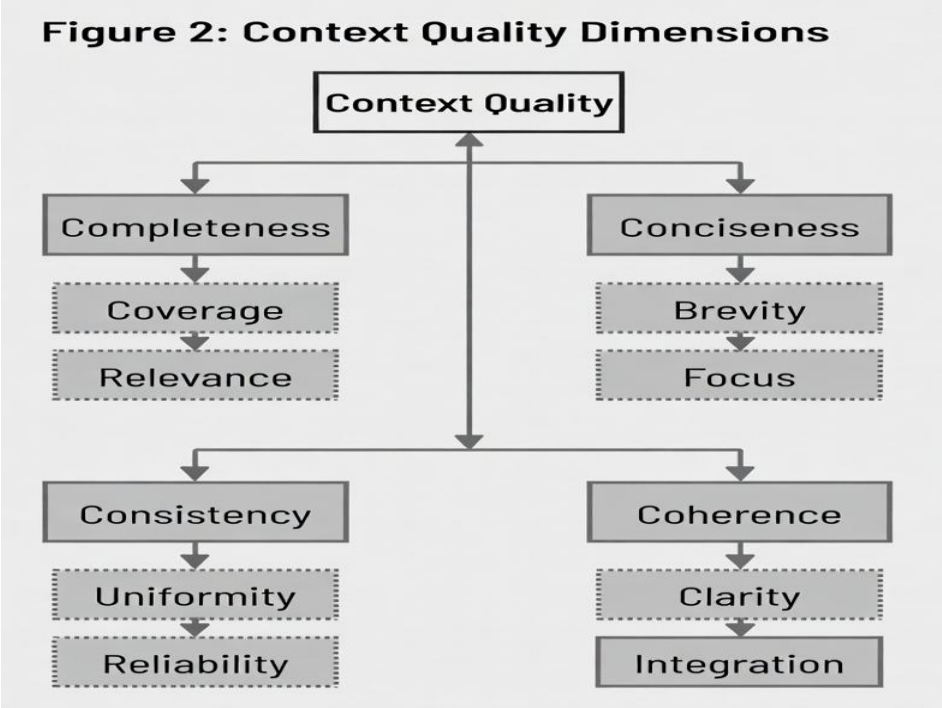

Fig 2: Context Quality Dimension

4.2 Context Quality Score (CQS)

CQS = $w_1 \cdot A + w_2 \cdot C + w_3 \cdot F + w_4 \cdot S + w_5 \cdot K + w_6 \cdot L$. A,C,F,S,K,L are normalized scores (0–1) for Accuracy, Completeness, Freshness, Structure, Conciseness, and Lineage. Illustrative Calculation (based on literature benchmarks [1], [6]): Assume weights tuned for code generation (0.25, 0.20, 0.15, 0.15, 0.15, 0.10). Observed scores: Accuracy=0.92, Completeness=0.88, Freshness=0.95, Structure=0.90, Conciseness=0.85, Lineage=0.98. Then CQS = 0.91. Thresholds (e.g., CQS ≥ 0.85) serve as pipeline gates.

5. ANALYTICAL FRAMEWORK AND SYNTHESIS

This study employs rigorous literature synthesis. Table 1 summarizes comparative impacts derived from the cited corpus.

Table 1: Comparative Impact of Context Approaches

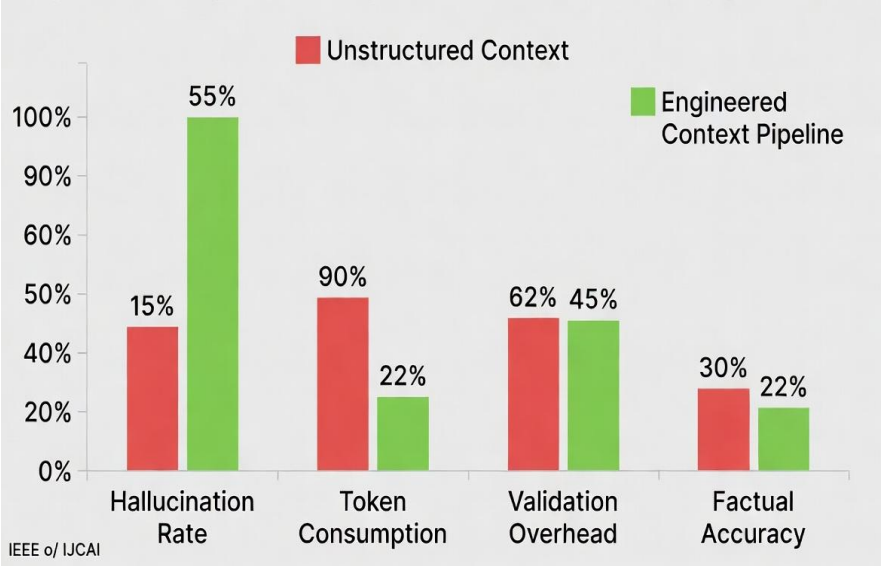

Fig 3: Comparative Impact Analysis

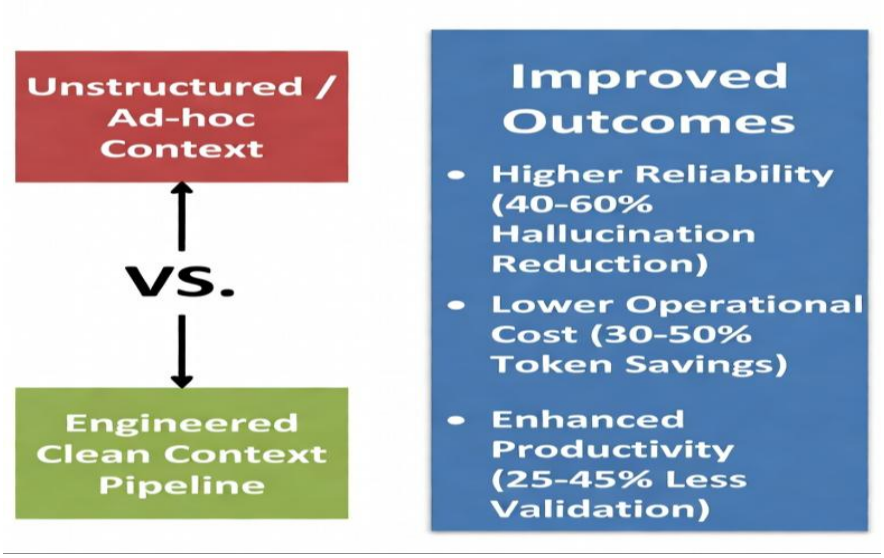

Fig 4: Impact of Context Engineering on LLM-Assisted Development

6. DISCUSSION OF IMPACTS

The integration of engineered context serves as a pivotal driver for reliability, directly mitigating factual drift by anchoring responses in structured data [1], [2]. This technical grounding is especially pronounced in code generation, where it significantly reduces the frequency of logical errors and invalid API calls [11], [16]. Beyond accuracy, the framework bolsters productivity by delivering cleaner inputs that minimize the need for manual validation cycles, thereby compounding the net time savings gained from raw generation speed [4]. From a fiscal perspective, cost efficiency is realized through strategic token optimization, which yields compounding reductions in inference expenses when deployed at scale [3]. These refinements also provide broader system benefits, including improved governance, cross-team consistency, and enhanced observability, ultimately establishing a robust foundation for future agentic workflows. It should be noted, however, that while these impacts are significant, they remain synthesis-based; future longitudinal studies in live production environments are required to quantify the exact effect sizes of these improvements.

7. CONCLUSION AND FUTURE DIRECTIONS

This paper establishes that clean context pipelines represent a foundational advancement in the evolution of LLM-assisted software development. While much of the current research ecosystem remains focused on model architectures, fine-tuning strategies, and larger training corpora, this work demonstrates that inference-time context quality is often the dominant determinant of reliability, operational efficiency, and economic viability in enterprise AI systems. By reframing context as a governed data product, the study positions data engineering as a central enabling discipline within modern LLMOps practices.

The proposed framework integrates established principles from data engineering including ingestion governance, semantic transformation, quality validation, lineage tracking, observability, and retrieval optimization into a unified architecture for context delivery. The synthesized literature indicates that these interventions can reduce hallucination rates by up to 40–60%, decrease token consumption by 30–50%, and significantly lower downstream human validation effort.

Importantly, these gains are achieved without altering the underlying foundation models, highlighting the disproportionate impact of high-quality contextual grounding.

The Context Quality Score (CQS) introduced in this work further contributes a practical governance mechanism for operationalizing context reliability in production environments. By quantifying dimensions such as accuracy, completeness, freshness, structure, conciseness, and lineage, organizations can establish measurable thresholds for trustworthy AI interactions and systematically monitor degradation over time.

The future scope of this research is substantial and extends across both academic and industrial domains. First, standardized benchmark datasets and evaluation frameworks for context quality in software engineering tasks remain largely absent. Establishing industry-wide benchmarks for context reliability, retrieval precision, hallucination frequency, and cost efficiency would significantly improve reproducibility and comparative evaluation across LLM systems.

Second, future systems are expected to evolve toward adaptive and self-optimizing context pipelines. Reinforcement learning techniques combined with human and AI feedback loops may enable dynamic context selection, automated summarization, intelligent token budgeting, and query-aware retrieval orchestration in real time. Such systems could continuously optimize relevance and cost trade-offs based on workload patterns and developer behavior.

Third, the rise of autonomous and agentic AI software engineering systems will further elevate the importance of trustworthy context engineering. Multi-agent development workflows involving planning agents, coding agents, testing agents, and governance agents will require highly structured, lineage-aware, and temporally synchronized context pipelines to ensure coordination accuracy and operational safety.

Fourth, integration with modern enterprise data architectures including data mesh, lakehouse platforms, vector databases, and real-time streaming systems presents a promising direction for scalable deployment. Future architectures may unify structured enterprise data, unstructured engineering artifacts, and live observability streams into continuously updated contextual knowledge fabrics for AI systems.

Fifth, emerging multimodal LLM ecosystems introduce new challenges and opportunities for context engineering. Future pipelines must support not only text and source code, but also diagrams, screenshots, architecture models, telemetry streams, and audio/video engineering artifacts. This expansion will require multimodal semantic indexing, cross-modal lineage tracking, and unified quality governance frameworks.

Another promising area involves predictive context quality analytics. Machine learning models could proactively identify low-quality or hallucination-prone contexts before inference occurs, enabling automated remediation, context regeneration, or retrieval fallback mechanisms. Such predictive observability could become a critical component of production-grade AI governance.

Finally, future empirical work should prioritize longitudinal enterprise case studies quantifying return on investment (ROI), developer productivity, operational savings, security improvements, and governance outcomes resulting from engineered context pipelines. Real-world production measurements across industries such as finance, healthcare, telecommunications, and cloud engineering would significantly strengthen the evidence base for context engineering as a mature discipline.

In conclusion, context engineering is rapidly emerging as a critical intersection between data engineering, information retrieval, and LLM operations. As organizations increasingly embed generative AI into software development lifecycles, the ability to deliver clean, governed, and cost-efficient context will become a defining capability for scalable and trustworthy AI systems. The future of reliable AI-assisted engineering will depend not only on better models, but on superior data engineering practices that ensure those models reason over accurate, relevant, and high-quality contextual information.

8. ACKNOWLEDGMENTS

The author thanks colleagues in data engineering and AI platforms for insightful discussions.

9. REFERENCES

[1] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2025. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. ACM Transactions on Information Systems, 43(2), Article 42. https://doi.org/10.1145/3703155 (arXiv:2311.05232).

[2] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of Hallucination in Natural Language Generation. ACM Computing Surveys, 55(12), Article 248. https://doi.org/10.1145/3571730 (arXiv:2202.03629).

[3] OpenAI. 2023. GPT-4 Technical Report. https://doi.org/10.48550/arXiv.2303.08774.

[4] Angela Fan, Beliz Gokkaya, Mark Harman, Mitya Lyubarskiy, Shubho Sengupta, Shin Yoo, and Jie M. Zhang. 2023. Large Language Models for Software Engineering: Survey and Open Problems. https://doi.org/10.48550/arXiv.2310.03533.

[5] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems, 33. https://doi.org/10.48550/arXiv.2005.11401.

[6] Carlo Batini and Monica Scannapieco. 2016. Data and Information Quality: Dimensions, Principles and Techniques. Springer. https://doi.org/10.1007/978-3-319-24106-7.

[7] Zhamak Dehghani. 2022. Data Mesh Principles and Logical Architecture. https://doi.org/10.48550/arXiv.2205.09750.

[8] Matei Zaharia, Andrew Chen, Aaron Davidson, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe, Fen Xie, and Corey Zumar. 2018. MLflow: A Unified Platform for Managing the Machine Learning Lifecycle. https://doi.org/10.1145/3183713.3190661.

[9] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: Simplified Data Processing on Large Clusters. Communications of the ACM, 51(1), 107–113. https://doi.org/10.1145/1327452.1327492.

[10] Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring How Models Mimic Human Falsehoods. Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 3214–3252. https://doi.org/10.18653/v1/2022.acl-long.229 (arXiv:2109.07958).

[11] Fang Liu, Yang Liu, Lin Shi, Zhen Yang, Li Zhang, Xiaoli Lian, Zhongqi Li, and Yuchi Ma. 2024. Beyond Functional Correctness: Exploring Hallucinations in LLM-Generated Code. https://doi.org/10.48550/arXiv.2404.00971.

[12] Sebastian Ruder. 2019. Neural Transfer Learning for Natural Language Processing. https://doi.org/10.48550/arXiv.1907.12484.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. https://doi.org/10.48550/arXiv.1706.03762.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://doi.org/10.48550/arXiv.1810.04805.

[15] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 38–45. https://doi.org/10.18653/v1/2020.emnlp-demos.6 (arXiv:1910.03771).

[16] Ankush Ramprakash Gautam. 2025. Impact of High Data Quality on LLM Hallucinations. International Journal of Computer Applications, 187(4), 35-39. DOI: 10.5120/ijca2025924909.