International Journal of Engineering Trends and Applications (IJETA) – Volume 13 Issue 3, May-Jun 2026

RESEARCH ARTICLE OPEN ACCESS

Application of RAG and RLHF in Modern Data Engineering Systems

Ratna Ram Tanwar[1], Abhishek Singh[2], Aryan Saxena[3], Aditya Kumar Singh[4], Ajay Choudhary[5]

[1]Assistant Professor, Department of Computer Science and Engineering, Global Institute of Technology, Jaipur, Rajasthan, India

[2,3,4,5] B.Tech Student, Department of CSE, Global Institute of Technology, Jaipur, Rajasthan, India

ABSTRACT: Data engineering has changed a lot between 2023 and 2026, moving from rigid ETL pipelines to more intelligent, self-healing systems. This shift is mainly driven by RAG and RLHF, where RAG provides real-time context and RLHF ensures accuracy and reliability. Modern tools are already using these ideas to automate tasks like SQL generation and pipeline management. However, challenges like evaluation issues and data drift still exist. Overall, the field is moving toward more autonomous systems with humans still playing an important role.

Keywords — RAG, RLHF, data engineering, ETL pipelines, self-healing systems, text-to-SQL, agentic systems, data pipelines, multimodal lakehouse, human-in-the-loop.

I. INTRODUCTION

Things in data engineering have changed quite a bit over the last few years (2023–2026). Traditional data pipelines that relied on rigid ETL (Extract, Transform, Load) processes and fixed directed acyclic graphs (DAGs) are slowly being replaced by more intelligent and self-healing systems. Manual pipelines are becoming too slow for modern requirements. This shift is mainly driven by two major AI techniques: Retrieval-Augmented Generation (RAG) and Reinforcement Learning from Human Feedback (RLHF). This work presents a detailed analysis of how these two approaches are being used in modern data engineering workflows. In simple terms, RAG helps language models pull in outside, real-time company data instead of just relying on what they already know, while RLHF is used to guide the model so its responses match what humans expect and keep the data quality in check. By looking at recent research, real industry use cases, and actual developer feedback from platforms like Reddit, this work tries to show how things are moving from simple assisted coding tools to more agent-like data engineering systems. It also talks about RAG and RLHF separately, and then how they work together in real architectures. Some real-world examples like Databricks Genie and dbt Copilot are included to make things more practical. At the same time, there are still many challenges in production, like what developers call “eval hell” and also the problem of knowledge drift. In the end, the report gives a brief idea about future trends, such as autonomous data pipelines, multimodal lakehouses, and why keeping humans in the loop (HITL) is still important

II. BACKGROUND

To truly understand how modern data systems are evolving, it is crucial to break down the mechanics of both RAG and RLHF. Neither of these is a single, out-of-the-box algorithm; rather, they are complex architectural patterns that solve different limitations of foundational AI models. You can think of them as two halves of a brain: RAG handles the short-term contextual memory, while RLHF handles the long-term behavioral instincts

A. RAG Concept

Retrieval-Augmented Generation (RAG) was introduced mainly to fix one big problem with LLMs they only know what they were trained on, up to a certain cutoff date. But in real companies, data keeps changing all the time, sometimes every minute. Continuously retraining a huge model just to keep it updated is not really practical or affordable for most organizations. So instead of retraining, RAG changes how the system works by dividing it into two steps: retrieval and generation.

The process usually begins with preparing the data. Unstructured data like PDFs, internal docs, wiki pages, or even Slack messages is broken into smaller parts, often called chunks. These chunks are then passed through an embedding model (like OpenAI embeddings or open-source ones such as BGE). The model converts text into numerical vectors, which basically represent the meaning of the text in a mathematical way. These vectors are stored in a vector database like Pinecone, Milvus, FAISS, or even PostgreSQL with pgvector.

When a query comes in, it is also converted into a vector using the same embedding model. Then the system searches for similar vectors, usually using cosine similarity, to find the most relevant chunks. After that, these chunks are added into the prompt as context, and the LLM generates a response based on that. Because of this, the model relies more on real data instead of its memory, which helps reduce hallucinations and improves accuracy. There are also different levels of RAG from simple ones that just add context, to more advanced setups with query rewriting, re-ranking, and better chunking

B. RLHF Concept

If RAG is about giving the model the right information, RLHF is about making sure the model behaves in the right way. It is one of the main reasons why modern chat models feel more natural, helpful, and less random. In data engineering, this becomes useful when you want structured outputs, like clean JSON, instead of long conversational answers. The RLHF process usually has three steps. First is Supervised Fine-Tuning (SFT), where the base model is trained on good examples of prompts and responses, so it learns the correct format. Second, a reward model is trained. Humans compare different outputs and rank them based on quality, safety, and correctness. The reward model learns from this and gives a score to outputs. Third, reinforcement learning (often using PPO) is applied. The model generates responses, gets scored by the reward model, and then updates itself to improve future responses.

But later, around 2024–2025, people started moving towards a simpler method called Direct Preference Optimization (DPO). The older RLHF setup was quite complex and heavy, especially because it needed multiple models and careful tuning. DPO removes the need for a separate reward model and instead learns directly from human preferences, like choosing which response is better. This makes the training process simpler and more efficient while still achieving good results.

Feature	**RLHF (with PPO)**	**Direct Preference Optimization (DPO)**
Pipeline Complexity	High	Low
Resource Efficiency	Low	High
Feedback Type	Multi-modal	Binary preferences only

III. ROLE OF RAG IN DATA ENGINEERING

Data engineering is mainly about managing state, schema, and context knowing where data is, what format it's in, and how it connects with other data. RAG fits well here because it allows AI systems to work with real context, almost like a smart data operator. One common use is in enterprise knowledge management. Instead of manually searching through docs or Slack, engineers can just ask questions like where a dataset is stored or how it is calculated, and get clear answers quickly. This saves a lot of time. RAG is also useful in text-to-SQL systems. Converting natural language into SQL is difficult if the model doesn't know the database schema. RAG helps

by retrieving relevant schema details and past queries, making the generated SQL more accurate and aligned with real business logic. However, using RAG in real systems is not easy. Data changes frequently, and keeping embeddings updated becomes a challenge, often leading to outdated results (knowledge drift). It also struggles with non-text data like tables or diagrams, which need extra processing before use. Even chunking the data properly is tricky too small loses context, too large creates confusion. Overall, while RAG is powerful, building it in production is still quite messy.

IV. ROLE OF RLHF IN DATA SYSTEMS

While RAG gives the model the right context and data, RLHF (and newer methods like DPO) focus on making the output reliable and correct. In data engineering, accuracy is critical. A model that sounds good but generates wrong SQL is basically useless. RLHF helps models focus more on correctness, proper execution, and performance instead of just sounding fluent. One major use is in text-to-SQL systems. Basic fine-tuning teaches models SQL syntax, but they often struggle with complex joins or nested queries. They might generate queries that look correct but fail when executed. RLHF improves this by rewarding queries that actually run correctly and penalizing wrong ones, helping the model learn real database logic.

There has also been a lot of discussion around PPO vs DPO. PPO was earlier considered better for complex tasks like SQL, while DPO was simpler and more stable. But recent improvements around 2025 changed this. By adding step-by-step reasoning (Chain-of-Thought) into DPO training, models started performing much better. This helped even smaller open-source models reach accuracy close to larger ones, making DPO more practical for real-world use. Beyond SQL, RLHF is also used in areas like data quality monitoring and anomaly detection. Instead of writing strict rules, systems can learn from simple human feedback (like marking something as correct or incorrect) and improve over time. It's also being used in entity resolution, where the system learns to match duplicate or messy records with very little labeled data, reducing a lot of manual effort.

V. INTEGRATION OF RAG AND RLHF

A. Architecture

A fully integrated system usually follows a dual-loop setup. In the first loop, RAG works like dynamic memory. It connects to things like data lakes, schema registries, and code repositories (e.g., GitHub). When a task comes in, like building a new ETL pipeline, the RAG system fetches useful context such as SQL patterns, past pipelines, and current table schemas. The second loop is the RLHF-tuned model. Since it has been trained with human feedback, it focuses more on accuracy and stability rather than creativity. It uses the retrieved data carefully, filters out irrelevant parts, and generates more reliable outputs. Some systems even train models to judge the quality of retrieved data before using it, which improves results further

.B. Pipelines and Workflows

One of the most useful applications of this setup is self-healing ETL pipelines. In traditional systems, even a small change like a data format or delimiter can break the pipeline and require manual fixing. With AI integration, a "try-heal-retry" approach is used. When something fails, the system collects logs, errors, and sample data, and passes it through RAG to find similar past issues. The RLHF-trained model then analyzes the problem and suggests a precise fix, usually in a structured format.

For example, it might detect that a CSV delimiter has changed and suggest updating a parameter in the code. The system tests this fix, and if it works, applies it automatically and restarts the pipeline. This happens quickly without human intervention. Overall, this kind of system moves data engineering from reactive fixes to more proactive and self-improving workflows.

VI. REAL WORLD USE CASE

The shift from research to real-world use has been very fast. By 2026, many major data platforms have started using RAG and RLHF directly in their products. This has moved the industry from basic AI copilots to more agent-like systems that can actually take actions.

One good example is Databricks Genie Code. It works like an AI assistant inside the workspace but goes beyond simple autocomplete. It uses RAG to understand company data, schemas, and policies. If a pipeline fails, for example due to a schema change, it can detect the issue, generate a fix, test it, and suggest it before a human even steps in. This shows how systems are becoming more self-healing. Another example is dbt Copilot. It helps in data transformation tasks by using RAG to understand the project's structure, models, and naming style. Engineers can generate SQL, documentation, and tests automatically. Since it is based on the actual project context, the output is more accurate and consistent, which reduces manual work a lot.

Tools like SQLMesh and Spice AI also show this trend. They allow users to interact with data pipelines using natural language. The system understands dependencies, checks the impact of changes, and runs validations automatically. This helps avoid common issues where one change breaks multiple downstream systems. Matillion Maia is another example, where AI is used to manage both batch and streaming pipelines. It can understand schemas, suggest transformations, and even detect data issues during processing. This makes it easier even for non-experts to build pipelines without writing complex code. Overall, these examples show that RAG and RLHF are no longer just research ideas they are already being used to build smarter, more automated data engineering systems.

VII. CHALLENGES AND TRENDS

Even though there has been a lot of progress, using LLMs in data engineering is still not easy in practice. Many developers say there is a big gap between research results and real production systems. One of the biggest issues is evaluation, often called "eval hell." Unlike normal software where things either pass or fail, it is hard to measure whether a RAG system is actually giving better results. Most evaluations depend on static datasets or manual checks, which don't work well in dynamic environments. Automated evaluation using LLMs also has its own problems, like bias toward longer answers.

Another challenge is latency and infrastructure cost. A typical RAG pipeline involves multiple steps embedding, retrieval, generation which adds delay. For real-time systems, even a few seconds of delay can be unacceptable. While long-context models try to solve this by putting everything into one prompt, they increase cost a lot and are not scalable.

On the RLHF side, the main problem is collecting good human feedback. Training models requires domain experts to review and rank outputs, which is expensive and time-consuming. Even with newer methods like DPO, this issue still exists. Also, trust is a major concern. Data systems require correct and consistent outputs, but AI models are probabilistic. A small mistake in SQL or metrics can lead to serious problems, so many teams are still cautious.

Looking ahead, the focus is shifting toward more autonomous systems. One major trend is the use of multiple AI agents instead of a single model. Different agents can handle tasks like monitoring data quality, detecting schema changes, or managing pipelines, and work together to fix issues automatically. This can reduce recovery time and make systems more efficient. Another trend is the rise of multimodal lakehouses, where structured and unstructured data can be handled together. This makes it easier for AI systems to gather context without relying on separate tools. At the same time, synthetic data generation is becoming important. Since human feedback is limited, companies are using AI to generate training data and improve smaller, specialized models. Overall, the future seems to be moving toward more automated, self-improving data systems, with humans playing more of a supervisory role.

V. CONCLUSION

The use of Retrieval-Augmented Generation (RAG) and Reinforcement Learning from Human Feedback (RLHF) has clearly changed how modern data engineering works. By combining RAG's ability to provide real-time context with RLHF's focus on accuracy and reliability, companies are building systems that are both smart and more stable. Today, many tools can automatically manage metadata, generate complex SQL queries with high accuracy, and even fix broken pipelines without human help. Still, there are challenges like high costs of collecting human feedback, delays in retrieval systems, and difficulties in proper evaluation. Even with these issues, the direction is quite clear. Data engineering is moving toward more agent-based systems where AI handles most of the repetitive work. As technologies like DPO and multimodal systems improve, access to data will become easier across organizations. In the end, this shift helps engineers spend less time fixing small issues and more time focusing on bigger design and decision-making problems.

ACKNOWLEDGMENT

The ideas and analysis shared in this work come from a mix of recent research papers, industry blogs, platform docs, and real developer discussions from 2023 to 2026. Overall, it shows how modern data engineering has evolved with contributions from the wider machine learning and data engineering community.

REFERENCES

[1] Mohsin, et al., "Online reinforcement learning-based RAG (OnRL-RAG)," arXiv:2504.02894, 2025.

[2] Databricks, "What is retrieval-augmented generation?," Databricks Blog, 2025.

[3] AWS, "What is reinforcement learning from human feedback?," Amazon Web Services, 2025.

[4] IBM, "Reinforcement learning from human feedback (RLHF)," IBM Think Topics, 2025.

[5] Red Hat, "Redefining development: retrieval-augmented generation (RAG) revolution," 2025.

[6] "RAG pain points and unsolved issues," Reddit Discussions (r/RAG), 2024–2025.

[7] Yuvraj Pratap Singh, Ratna Ram Tanwar, Kritika, "Zero-Trust Security Models in Enterprise Networks: Principles, Implementations and Future Resilience", International Journal of Global Research in Science and Technology (IJGRST), Vol. 10, pp. 80-83, 2025.

[8] Harshita Chandrawat, Hitesh Kumar, Ishwar Verma, Manju Mathur, Ratna Ram Tanwar, "Vegetable Crop Disease Detection Using Machine Learning and Artificial Intelligence", Implementations and Future Resilience", International Journal of Global Research in Science and Technology (IJGRST), Vol. 10, pp. 199-202, 2025.

[9] Kritika Paliwal, Laxmikant Vashishtha, Ratna Ram Tanwar, "Big Data Analytics in Modern Engineering and Industry", International Journal of Global Research in Science and Technology (IJGRST), Vol. 10, pp. 326-330, 2025.

[10] M. K. Sain and N. Sharma, "A study of research issues and challenges of big data analytics," Journal of Advances and Scholarly Researches in Allied Education, vol. 16, no. 5, pp. 1699–1707, 2019.

[11] G. Sharma, N. Hemrajani, S. Sharma, A. Upadhyay, Y. Bhardwaj, and A. Kumar, "Data management framework for IoT edge-cloud architecture for resource-constrained IoT application," Journal of Discrete Mathematical Sciences and Cryptography, vol. 25, no. 4, pp. 1093–1103, 2022.

[12] Manish Jha, "A Study of ISA Server for Providing Fast Internet Access with a Single Proxy", SGVU Journal Of Engineering & Technology, Vol. 1, Issue. 1, pp. 15-18, 2015.

[13] N. Sharma, Dr. M. K. Sain, "An OOHI Analysis Approach for Distributed Data Store and Complex Event Processing of Big Data", Journal of Information and

Computational Science, Vol. 11, Issue. 10, pp. 375-383, 2021.

[14] M. K. Jha, "Recent Trends and Emerging Applications of the Internet of Things: Transforming the Way We Live and Work", International Journal of Engineering Trends and Applications (IJETA), Vol. 12, Issue. 4, pp. 239-244, 2025.

[15] Manish Kumar Jha, Siddhi Agarwal, Vishakha Kabra, "Artificial Intelligence at Work Transforming Industries and Redefining the Workforce Landscape", International Journal of Engineering Trends and Applications, Vol. 12, Issue. 4, pp. 416-424, 2025.

[16] N. Sharma, "An analytical study of distributed data store using big data analysis technique," Research Methods, Imparc, 2019.P. Upadhyay, K. K. Sharma, R. Dwivedi and P. Jha, "A Statistical Machine Learning Approach to Optimize Workload in Cloud Data Centre," 2023 7th International Conference on Computing Methodologies and Communication (ICCMC), pp. 276-280, 2023.

[17] Matillion, "ETL automation and Maia," Matillion Learning, 2025.

[18] Nweke, B., "Building a self-healing data pipeline that fixes its own Python errors," Towards Data Science, 2026.

[19] Liu, et al., "Uncovering the impact of chain-of-thought reasoning for direct preference optimization," ACL, 2025.

[20] Shankar, J., "Human-in-the-loop: the architecture pattern behind reliable AI," Medium, 2025.

[21] DeFauw, et al., "Build a RAG data ingestion pipeline," AWS Big Data Blog, 2024.

[22] Tobiko Data, "SQLMesh and Spice AI," Tobiko Data Blog, 2025.

[23] "RLHF vs DPO for SQL generation opinions," Reddit Discussions (r/dataengineering), 2024–2025.

[24] Arbisoft, "RLHF vs DPO: a closer look into the process," 2025.