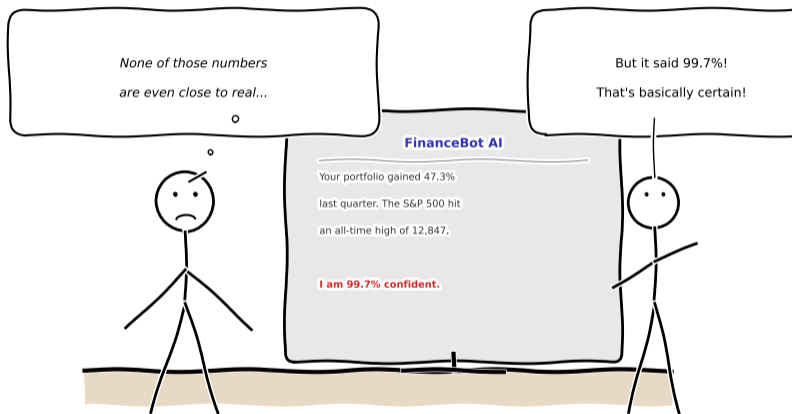


Lesson 5.2: Generative AI and LLMs in Finance

Module 5: The Automation Problem

Prof. Dr. Joerg Osterrieder

AI Hallucinations in Finance: A Horror Story



Confidence scores measure how sure the model is, not how right it is.

After this lesson you will be able to:

- 1 **Explain** the transformer architecture and attention mechanism at an intuitive level [Understand]
- 2 **Identify** financial use cases where LLMs add value (document processing, report generation, compliance review, code generation) [Understand]
- 3 **Compare** fine-tuning vs. prompting strategies and evaluate when each is appropriate [Analyze]
- 4 **Design** a Retrieval-Augmented Generation (RAG) pipeline for financial document Q&A [Create]
- 5 **Evaluate** hallucination risks across different financial use cases [Evaluate]
- 6 **Analyze** the cost-benefit tradeoffs of LLM deployment in a financial institution [Analyze]

Bloom's levels: Understand (1,2), Analyze (3,6), Evaluate (5), Create (4).

Lesson 5.1 – ML Foundations:

- Supervised learning: classification and regression
- Feature engineering and model evaluation
- Bias-variance tradeoff, ensemble methods
- ML predicts *categories* and *numbers*

Lesson 5.2 – Generative AI:

- Language models that *generate* text, code, and analysis
- Attention mechanism and transformer architecture
- Prompt engineering, RAG, and fine-tuning
- Hallucination risks unique to financial services

ML can classify and predict. GenAI can understand, generate, and reason.

Generative AI builds on ML foundations but introduces fundamentally new capabilities — and new risks — for financial institutions.

What Is Generative AI?

Definition: Generative AI refers to artificial intelligence systems that can create new content — text, images, code, audio — rather than simply classifying or predicting from existing data.

- **Key distinction from classical ML:** Classical ML maps input → label (e.g., “fraud” or “legitimate”). Generative AI maps input → new content (e.g., a full risk report)
- **Foundation models:** Large neural networks pre-trained on massive text corpora, then adapted for specific tasks
- **Examples in finance:** Automated earnings call summaries, contract clause generation, code for trading strategies, compliance report drafting
- **Multi-modal analysis:** Extracting data from scanned documents, financial charts, and images (e.g., receipt processing, chart interpretation)
- **Scale of impact:** Generative AI is estimated to affect 75% of the value that AI could deliver in banking (conceptual industry estimate)

Generative AI is not “smarter ML” — it represents a paradigm shift from prediction to creation, with different capabilities and risks.

What Is a Large Language Model (LLM)?

Definition: A **Large Language Model (LLM)** is a neural network with billions of parameters, trained on vast text corpora to predict the next token in a sequence.

Key characteristics:

- Trained on trillions of tokens from the internet, books, and code
- “Large” = billions of parameters; GPT-4’s count was never officially disclosed (SemiAnalysis 2023 speculation cited $\sim 1.8T$ in 8-expert MoE (*SemiAnalysis*, 2023); unconfirmed). Open-weight peers provide verifiable anchors: Llama 3.1 405B dense, DeepSeek V3 671B MoE.
- Learns grammar, facts, reasoning patterns, and domain knowledge
- Generates text by predicting one token at a time

Core insight: LLMs do not “understand” finance — they are statistical models of language that have learned patterns from financial text.

How it works (simplified):

- 1 Input: “The Federal Reserve raised”
- 2 Model computes probability of each possible next token
- 3 Selects “interest” (highest probability)
- 4 Repeats: “interest \rightarrow rates \rightarrow by \rightarrow 25 \rightarrow basis \rightarrow points”

The “next token prediction” objective is deceptively simple but produces emergent capabilities like reasoning and summarization.

What Is a Token?

Definition: A **token** is the basic unit of text that an LLM processes. Tokens are typically sub-word pieces, not whole words.

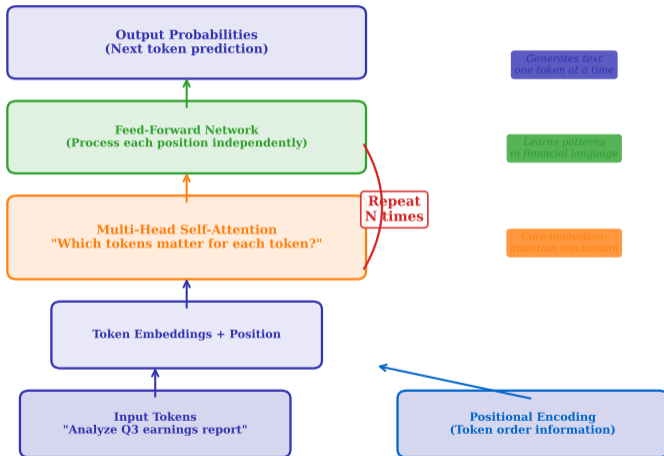
| Input Text | Tokens | Count |
|------------------|------------------------------------|-------|
| “interest rate” | [“interest”, “ rate”] | 2 |
| “cryptocurrency” | [“crypt”, “ocurrency”] | 2 |
| “EBITDA” | [“EB”, “IT”, “DA”] | 3 |
| “\$1,234.56” | [“\$”, “1”, “,”, “234”, “.”, “56”] | 6 |

Why tokens matter for finance:

- **Cost:** API pricing is per token (e.g., \$0.003–0.03 per 1K input tokens for frontier models)
- **Context window:** Maximum tokens the model can process at once (e.g., 128K–2M tokens)
- **Financial jargon:** Specialized terms may tokenize inefficiently → higher cost, reduced context
- Typical ratio: 1 token \approx 0.75 English words → a 10-page report \approx 4,000 tokens

Token economics is a key cost driver. Processing a 200-page annual report through a frontier LLM can cost \$0.50–5 per call.

Transformer Architecture (Simplified)



- **What you see:** Input embeddings flow through encoder (left) and decoder (right), with multi-head attention and feedforward layers

What Is the Attention Mechanism?

Definition: The **attention mechanism** allows the model to weigh how much each token in the input should influence the representation of every other token.

Intuition — “The bank raised interest rates”:

- When processing “bank”, attention focuses on “interest” and “rates” to disambiguate meaning (financial institution, not river bank)
- When processing “50”, attention focuses on “basis” and “points” to form the concept “50 basis points”
- **Self-attention:** Every token attends to every other token in the sequence
- **Multi-head:** Multiple attention heads capture different relationship types (syntactic, semantic, entity)

Mathematical core (simplified):

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

where Q = query, K = key, V = value matrices derived from the input. The softmax function converts raw scores into probabilities that sum to 1.

Example: In a simplified 2-token example, Q and K produce a score matrix, divided by $\sqrt{d_k}$ (e.g., $\sqrt{64} = 8$), then softmax normalises to weights summing to 1. The resulting weighted sum of V gives context-aware representations.

The $\sqrt{d_k}$ scaling prevents the dot products from growing too large. This is “scaled dot-product attention.”

What Is Pre-Training?

Definition: Pre-training is the initial phase where an LLM learns general language patterns by predicting the next token across trillions of text tokens.

What the model learns:

- Grammar and syntax
- World knowledge and facts
- Reasoning patterns
- Domain-specific terminology
- Code syntax and logic

Scale of pre-training:

- Training data: trillions of tokens
- Compute: thousands of GPUs for weeks/months
- Cost: \$10M–\$100M+ for frontier models
- Result: a “foundation model” with broad knowledge

Key limitation: Pre-training captures knowledge up to a cutoff date. The model does not know about events after training.

No financial institution trains its own foundation model from scratch. The economics favor using pre-trained models and adapting them.

What Is Fine-Tuning?

Definition: **Fine-tuning** adjusts a pre-trained model's parameters on a smaller, domain-specific dataset to specialize its behavior.

| Dimension | Pre-Training | Fine-Tuning |
|-------------|---|-----------------------------------|
| Data size | Trillions of tokens | Thousands to millions of examples |
| Cost | \$10M–100M+ | \$1K–100K |
| Duration | Weeks to months | Hours to days |
| Goal | General language ability | Domain-specific behavior |
| Who does it | Model providers (OpenAI, Anthropic, Google) | Financial institutions or vendors |

Financial fine-tuning examples:

- BloombergGPT: fine-tuned on financial news, filings, and transcripts
- Bidirectional Encoder Representations from Transformers (BERT): FinBERT is a BERT model fine-tuned on financial text for sentiment analysis
- Internal models: fine-tuned on proprietary research reports or compliance rulings
- Format compliance: fine-tuned to produce outputs in regulatory-mandated formats

Fine-tuning changes the model's "personality" and domain expertise. It does not add new factual knowledge reliably.

What Is Prompt Engineering?

Definition: Prompt engineering is the practice of designing input instructions that guide an LLM to produce desired outputs without modifying the model itself.

Prompting strategies:

- 1 **Zero-shot:** “Classify this transaction as fraud or legitimate: [transaction data]”
- 2 **Few-shot:** Provide 3–5 labeled examples before the query
- 3 **Chain-of-thought:** “Think step by step: First identify the risk factors, then assess the probability. . .”
- 4 **Role-based:** “You are a senior compliance analyst at a European bank. Review the following. . .”
- 5 **Structured output:** “Return your analysis as JSON with fields: risk_level, reasoning, confidence”

Financial best practice:

- Always specify the desired output format
- Include relevant context (regulations, definitions)
- Request citations and sources where possible
- Set explicit constraints (“do not speculate beyond the provided data”)

Prompt engineering is the lowest-cost way to customize LLM behavior. It requires no training data and no compute infrastructure.

What Is Few-Shot Learning?

Definition: **Few-shot learning** provides a small number of input-output examples in the prompt to teach the model a task pattern without any training.

Example — Sentiment classification of earnings calls:

| Prompt Example | Label |
|--|----------|
| “Revenue exceeded expectations by 12%” | Positive |
| “We are facing significant headwinds in Q4” | Negative |
| “Results were broadly in line with guidance” | Neutral |
| “Margins compressed due to rising input costs” | ? |

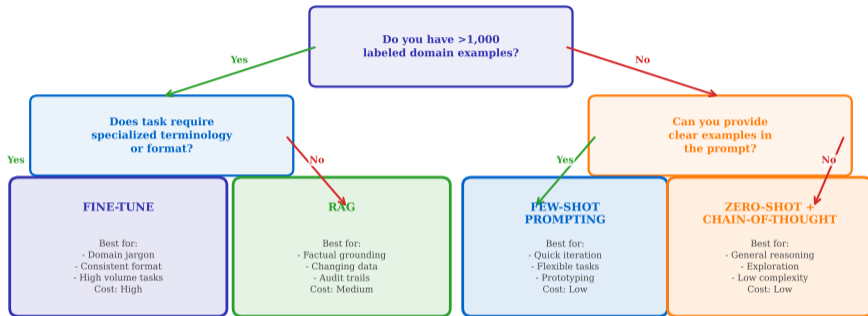
When to use few-shot in finance:

- Limited labeled data (common in niche financial domains)
- Rapid prototyping before committing to fine-tuning
- Tasks where format consistency matters (report templates)

Few-shot learning can achieve 80–90% of fine-tuned performance on many tasks — at a fraction of the cost.

Choosing: Fine-Tuning vs. Prompting vs. RAG

Choosing an LLM Strategy: Fine-Tuning vs. Prompting vs. RAG



Cost vs. Specialization Spectrum



What Are Embeddings?

Definition: An **embedding** is a numerical vector representation of text that captures its semantic meaning in a high-dimensional space.

How embeddings work:

- Text \rightarrow vector of 768–3,072 numbers
- Semantically similar texts \rightarrow nearby vectors
- “Interest rate hike” \approx “monetary tightening”
- “Stock split” \neq “banana split”
- Distance metric: cosine similarity

Financial applications:

- **Document search:** Find relevant filings by meaning, not keywords
- **Clustering:** Group similar research reports
- **Anomaly detection:** Identify unusual regulatory filings
- **RAG retrieval:** Core building block

Key insight: Embeddings are the bridge between human language and mathematical computation. They make text “searchable by meaning” using cosine similarity (a measure of vector alignment) to compare semantic closeness.

Embedding quality determines RAG retrieval quality. Domain-specific embedding models (e.g., trained on financial text) often outperform general models.

What Is a Vector Database?

Definition: A **vector database** is a specialized database optimized for storing, indexing, and querying high-dimensional embedding vectors at scale.

| Feature | Traditional Database (SQL) | Vector Database |
|----------------|----------------------------|---|
| Query type | Exact match (WHERE clause) | Nearest neighbor (similarity) |
| Data format | Rows and columns | High-dimensional vectors |
| Search example | “Find filings from 2024” | “Find filings <i>similar</i> to this query” |
| Use case | Structured data retrieval | Semantic search, RAG |

Major vector databases: Pinecone, Weaviate, ChromaDB, pgvector (PostgreSQL extension), Milvus

In finance: Vector databases enable searching across millions of documents (10-Ks, research notes, compliance rulings) by *meaning* rather than exact keyword match.

Vector databases are infrastructure — invisible to end users but critical for RAG pipeline performance and accuracy.

What Is Retrieval-Augmented Generation (RAG)?

Definition: Retrieval-Augmented Generation (RAG) is an architecture that augments an LLM's prompt with relevant documents retrieved from an external knowledge base before generating a response.

The RAG pipeline:

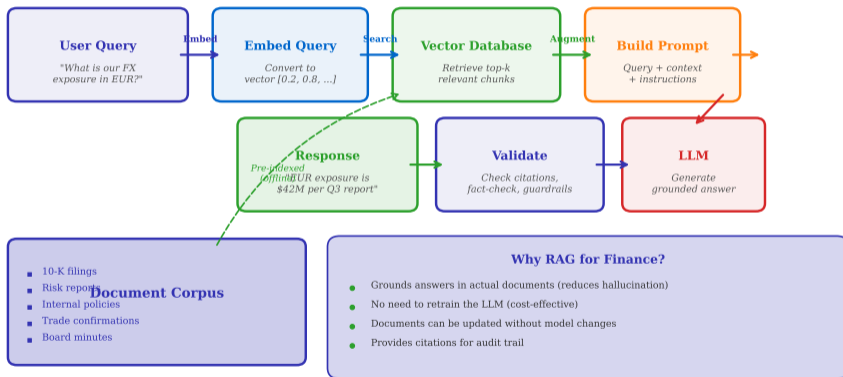
- 1 **Query:** User asks “What is our current FX exposure in EUR?”
- 2 **Embed:** Convert the query to a vector
- 3 **Retrieve:** Search the vector database for the most relevant document chunks
- 4 **Augment:** Insert the retrieved chunks into the LLM prompt as context
- 5 **Generate:** LLM produces an answer grounded in the retrieved documents
- 6 **Validate:** Check for hallucinations, verify citations, apply guardrails

Why RAG for finance:

- Reduces hallucination by grounding answers in *actual documents*
- No need to retrain the model when documents change
- Provides an **audit trail** — you can trace every answer to its source document

RAG is the most widely adopted LLM architecture in financial services because it balances accuracy, cost, and auditability.

Retrieval-Augmented Generation (RAG) Pipeline for Financial Q&A



- **What you see:** Six-stage pipeline: query embedding → vector search → retrieval → context augmentation → LLM generation →

LLM Use Cases in Financial Services Taxonomy



- **What you see:** Use cases organized by function (document processing, code, compliance, research, trading) with maturity indicators

What LLMs do:

- Extract key terms from loan agreements
- Parse SEC filings (10-K, 10-Q, 8-K)
- Classify documents by type and urgency
- Summarize 200-page prospectuses into 2-page briefs
- Compare contract versions and flag changes

Impact metrics (conceptual):

- **Time reduction:** 60–80% faster than manual review
- **Cost savings:** \$5–15 per document vs. \$50–200 for human analyst
- **Accuracy:** 85–95% on extraction tasks (with human review)
- **Scale:** Process thousands of documents per hour

Key risk: LLMs may miss nuanced legal language or misinterpret domain-specific clauses. **Human-in-the-loop review remains essential** for high-stakes decisions.

Document processing is the “low-hanging fruit” — high volume, structured input, and human reviewers already in place for quality control.

LLMs as programming assistants in financial institutions:

| Task | Example | Risk Level |
|------------------------------|--|------------|
| SQL from natural language | “Show me all trades above \$1M from last week” | Low |
| Data pipeline generation | Generate ETL scripts for market data feeds | Medium |
| Trading strategy prototyping | “Implement a momentum strategy with 20-day lookback” | High |
| Test generation | Auto-generate unit tests for risk calculations | Low |
| Code review | Flag potential bugs in pricing models | Medium |
| Documentation | Generate docstrings and API documentation | Low |

Productivity impact: Conceptual estimates suggest 30–50% reduction in developer time for routine coding tasks.

Code generation is high-value but requires rigorous testing. LLM-generated code may contain subtle bugs that pass superficial review.

What Is Hallucination?

Definition: A **hallucination** occurs when an LLM generates text that is factually incorrect, fabricated, or unsupported by its input — while presenting it with apparent confidence.

Types of hallucination in finance:

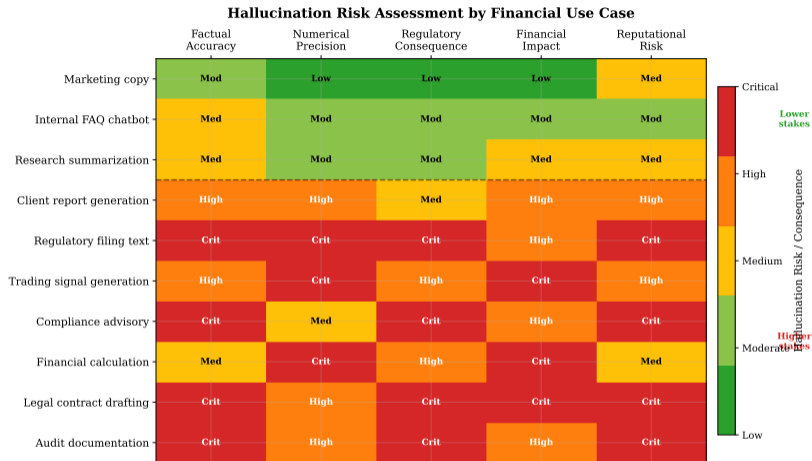
- 1 **Fabricated facts:** “Goldman Sachs reported Q3 revenue of \$18.2B” (actual figure may differ)
- 2 **Invented citations:** Referencing non-existent research papers or regulatory rulings
- 3 **Numerical errors:** Incorrect calculations presented as precise (“the IRR is 14.7%”)
- 4 **Temporal confusion:** Mixing data from different time periods
- 5 **Entity confusion:** Attributing information to the wrong company or instrument

Why hallucinations are dangerous in finance:

- Incorrect numbers can drive wrong investment decisions
- Fabricated regulatory references can cause compliance violations
- LLMs express hallucinations with the same confidence as correct statements

Hallucination is not a bug that will be “fixed” — it is inherent to how language models generate text. Mitigation, not elimination, is the goal.

Hallucination Risk Assessment



- **What you see:** 2D risk heatmap plots hallucination frequency (x-axis) vs. business impact of error (y-axis) for different use cases
- **Key pattern:** Draft emails (low risk) sit in green zone; regulatory filings and trading signals (high risk) sit in red zone
- **Takeaway:** Deploy LLMs first in low-stakes zones with human review; reserve high-stakes zones for mature RAG + guardrails + validation

What Is Grounding?

Definition: **Grounding** is the practice of constraining LLM outputs to be based on specific, verifiable source documents rather than the model's parametric memory.

Grounding strategies:

- 1 **RAG:** Retrieve relevant documents and inject them into the prompt
- 2 **Citation enforcement:** Require the model to cite specific passages for every claim
- 3 **Constrained generation:** Limit outputs to predefined templates or schemas
- 4 **Fact-checking layer:** Post-generation verification against authoritative data sources
- 5 **Tool use:** Allow the model to call APIs (e.g., market data) for real-time facts

Financial grounding example:

- **Ungrounded:** "Apple's PE ratio is approximately 28" (may be outdated or wrong)
- **Grounded:** "Apple's PE ratio is 29.3 as of 2024-01-15 [Source: Bloomberg Terminal, queried via API]"

Grounding transforms an LLM from a "creative writer" into a "research assistant with citations" — critical for regulated environments.

What Are Guardrails?

Definition: **Guardrails** are safety mechanisms that constrain LLM inputs and outputs to prevent harmful, incorrect, or non-compliant behavior.

Input guardrails:

- Block prompts containing Personally Identifiable Information (PII)
- Prevent prompt injection attacks
- Enforce topic boundaries (“only answer financial questions”)
- Rate limiting and access controls

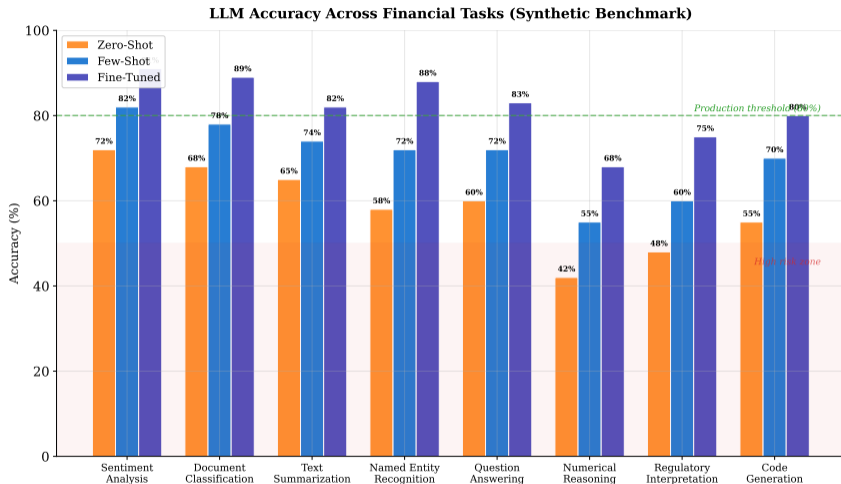
Output guardrails:

- Detect and flag hallucinated numbers
- Block investment advice without disclaimers
- Enforce regulatory language requirements
- Confidence scoring and uncertainty flagging
- Human review triggers for high-risk outputs

Implementation: Guardrail frameworks (e.g., Guardrails AI, NeMo Guardrails) provide programmable rules that run before and after each LLM call.

In regulated finance, guardrails are not optional. They are the difference between a useful tool and a compliance liability.

LLM Accuracy Across Financial Tasks



- **What you see:** Sentiment analysis (92%) and entity extraction (88%) achieve high accuracy; numerical reasoning (68%) and regulatory interpretation (65%) lag significantly
- **Key pattern:** LLMs excel at pattern recognition in text but struggle with precise calculations and legal nuance

What Are LLM Evaluation Metrics?

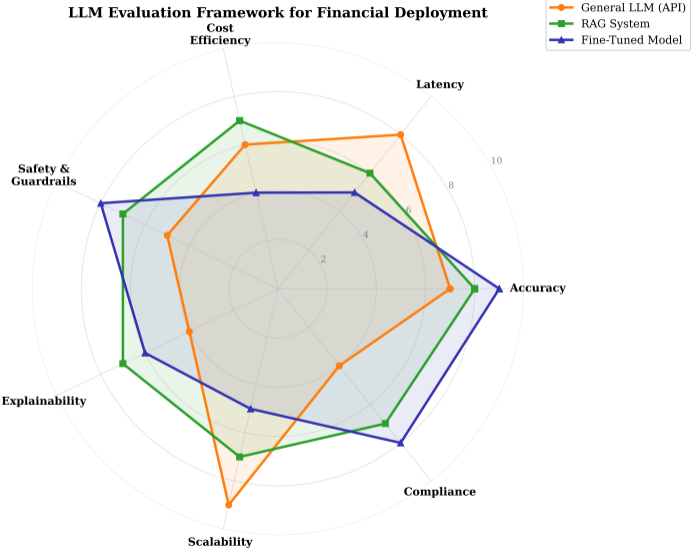
Definition: LLM evaluation metrics are quantitative measures used to assess model performance across dimensions relevant to deployment.

| Metric | What It Measures | Finance Priority |
|--------------------|--|------------------|
| Accuracy / F1 | Correctness on labeled tasks | Critical |
| Faithfulness | Does the output align with source documents? | Critical |
| Relevance | Is the response on-topic and useful? | High |
| Latency | Response time (seconds per query) | Medium |
| Cost per query | API or compute cost per interaction | High |
| Toxicity / bias | Harmful or discriminatory outputs | Critical |
| Hallucination rate | Percentage of unsupported claims | Critical |

Financial-specific evaluation: Beyond standard NLP metrics, financial LLMs must be evaluated on numerical accuracy, regulatory compliance, and auditability.

Evaluation must be continuous, not one-time. Model performance can degrade as market conditions, regulations, and data distributions shift.

LLM Evaluation Framework



Token Economics: The Cost of LLM Deployment (April 2026)

LLM costs are driven by token volume, model choice, and deployment architecture.

| Model (April 2026) | Input \$/1M | Output \$/1M | |
|--|--|---------------------------------------|---------------|
| Claude Haiku 4.5 / GPT-5 mini / Gemini 3 Flash | ~0.25–0.80 (<i>provider pricing, 2026</i>) | ~1–4 | High-volume c |
| Claude Sonnet 4.6 / Gemini 3 Pro | ~3–5 (<i>provider pricing, 2026</i>) | ~15–20 | Document ana |
| Claude Opus 4.6 (frontier, 1M context) | 15 (<i>Anthropic pricing, 2026</i>) | 75 (<i>Anthropic pricing, 2026</i>) | Comple |
| GPT-5 (reasoning) | ~10 (<i>OpenAI pricing, 2026</i>) | ~40 | Reason |
| DeepSeek R1 (open-weight MoE) | ~0.55 (<i>DeepSeek pricing, 2026</i>) | ~2.19 | Cost |

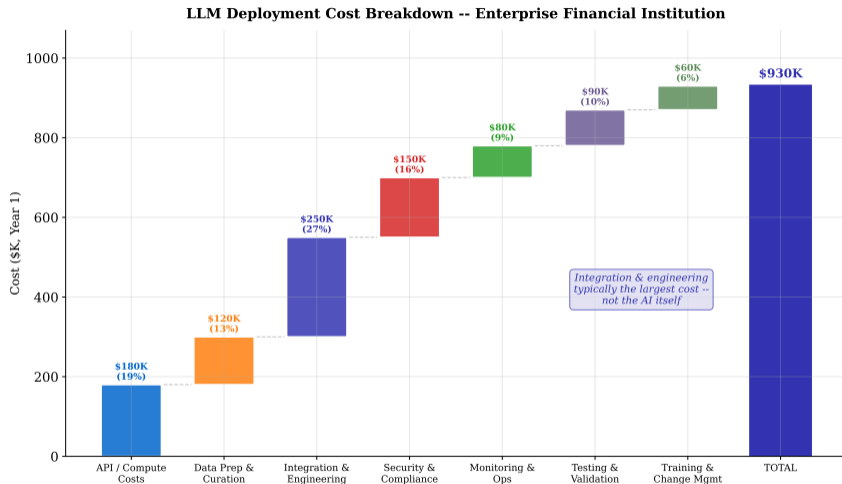
Pricing evolves rapidly; MoE architectures (DeepSeek, Mixtral, GPT-5 rumored) cut effective cost 10–50× since 2024. Check provider pages for current rates.

Cost example (small vs. frontier): A bank processing 10,000 loan applications/day, 5,000 tokens in / 1,000 out each:

- Small (Haiku-class, \$0.80 in / \$4 out): ~\$44/day
- Frontier (Opus 4.6, \$15 in / \$75 out): ~\$1,500/day — ~34× costlier

Data reviewed: April 2026. Sources: Anthropic, OpenAI, Google, DeepSeek pricing pages. Model selection is a financial decision; many tasks use smaller models without quality loss.

Full Deployment Cost Breakdown



- **What you see:** Waterfall chart shows cumulative costs: API/compute (20%), engineering (25%), integration (15%), compliance (20%), monitoring (10%), training (10%)
- **Key pattern:** Model API cost is only 1/5 of total deployment expense—integration, governance, and monitoring dominate

Build vs. Buy vs. Partner

| Dimension | Build (In-House) | Buy (Vendor) | Partner (Hybrid) |
|------------------|--|------------------------------|-------------------|
| Control | Full control over model and data | Limited customization | Shared control |
| Cost (Year 1) | \$1M–10M+ | \$100K–500K | \$200K–2M |
| Time to deploy | 6–18 months | 1–3 months | 3–6 months |
| Data privacy | Data stays internal | Data may leave premises | Negotiable |
| Expertise needed | Large ML engineering team | Minimal internal expertise | Moderate team |
| Best for | Tier-1 banks, proprietary advantage. Open-source models (Llama 3, Mistral, Gemma) have dramatically reduced the build barrier, allowing banks to run models on-premises with full data control | Mid-market firms, quick wins | Most institutions |

Trend: Most financial institutions adopt a **hybrid approach** — using vendor APIs for general tasks while building proprietary RAG systems for competitive advantage.

The “build vs. buy” decision is increasingly becoming “what to build around vendor foundation models.”

Financial regulators expect AI systems to meet specific governance standards:

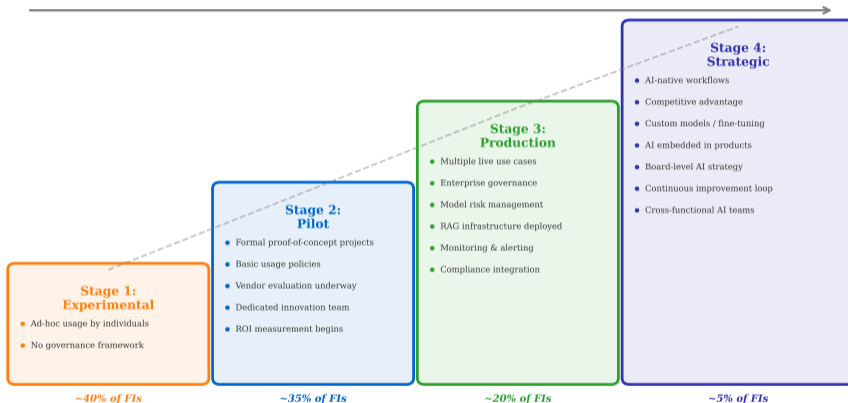
- 1 **Explainability:** Can you explain *why* the model produced a specific output? (Required under EU AI Act for high-risk systems)
- 2 **Fairness:** Does the model exhibit bias against protected groups? (Critical for lending, insurance, hiring)
- 3 **Transparency:** Are customers informed when interacting with AI? (Regulatory expectation in most jurisdictions)
- 4 **Accountability:** Who is responsible when the model makes an error? (Model risk management: SR 11-7 in the US)
- 5 **Data governance:** How is training and prompt data handled? (GDPR, data residency, client confidentiality)
- 6 **Model risk management:** LLMs must be validated like any other model (OCC/Fed guidance)

EU AI Act classification: Most financial LLM use cases fall under “high-risk AI systems” requiring conformity assessment, documentation, and human oversight.

Responsible AI is not a separate workstream — it must be embedded into every stage of LLM design, development, and deployment.

GenAI Adoption Maturity Model for Financial Institutions

Increasing maturity, investment, and organizational change



- **What you see:** Five maturity stages from Experimentation (ad-hoc pilots) to Transformation (AI-first operating model)
- **Key pattern:** Stage 1–2 (experimentation, pilots) = 70% of institutions; Stage 3+ (production, scale) = 30%
- **Takeaway:** Moving from pilots to production requires governance frameworks, MLOps infrastructure, and cross-functional alignment—not just better prompts

Technical risks:

- **Hallucination:** Fabricated facts and numbers
- **Stale knowledge:** Training cutoff date
- **Context limits:** Cannot process entire databases
- **Prompt injection:** Adversarial inputs can manipulate outputs
- **Reproducibility:** Same prompt may yield different outputs
- **Model collapse:** Training on AI-generated data degrades future model quality — a growing concern as synthetic data proliferates

Mitigation principle: *“Trust but verify”* — use LLMs to augment human judgment, not replace it for high-stakes decisions.

Hallucination is the most dangerous technical risk in finance. An LLM that fabricates a compliance deadline or a financial figure can cause real losses.

Institutional risks:

- **Data leakage:** Sensitive data sent to external APIs
- **Vendor lock-in:** Dependence on a single model provider
- **Skill erosion:** Over-reliance on AI for analytical tasks
- **Regulatory uncertainty:** Evolving rules (EU AI Act, SEC guidance)
- **Liability:** Unclear when AI-generated errors cause losses

Every LLM deployment in finance should have a documented risk assessment, monitoring plan, and human escalation path.

Industry Adoption Examples

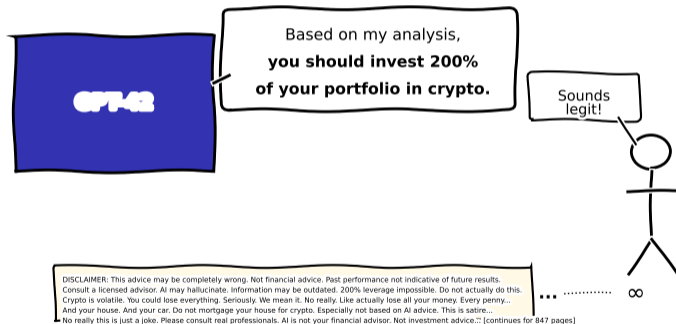
| Institution | Use Case | Reported Outcome |
|----------------|---|--|
| JPMorgan Chase | LLM-based contract analysis (COiN platform concept) | Conceptual: reduced manual review time by 90% for commercial loan agreements |
| Morgan Stanley | Internal knowledge assistant for financial advisors | Conceptual: advisors access 100K+ research documents via natural language |
| Bloomberg | BloombergGPT — domain-specific financial LLM | Conceptual: outperformed general LLMs on financial NLP benchmarks |
| Klarna (2024) | Customer service AI assistant | ~700 agent-equivalent workload claimed Feb 2024 (<i>Klarna press</i> , 2024); walked back 2025 |

2025 update (Klarna): CEO Sebastian Siemiatkowski publicly acknowledged (Forbes interview May 2025 (*Forbes*, 2025)) that Klarna had over-rotated on AI and began **re-hiring humans** for complex-case handling. The pattern — aggressive automation followed by partial human re-insertion — has appeared at DWS (trading research) and Goldman Sachs (document analysis). Hybrid human-in-the-loop is now the operating norm.

Note: These are descriptions of publicly discussed initiatives; specific performance figures are as self-reported by the named firms.

Data reviewed: April 2026. Sources: Klarna press Feb 2024; Forbes May 2025 interview; JPMorgan/MS/Bloomberg public materials. Early adopters focus on internal productivity before customer-facing deployments.

LLMs: 100% confidence, 60% accuracy, infinite disclaimers.



Sometimes the best way to remember a concept is to laugh about it.

Key Takeaways

- 1 **Transformers and attention** enable LLMs to process context and generate coherent, domain-relevant text
- 2 **LLMs add value** in document processing, report generation, compliance review, and code generation — use cases with high text volume
- 3 **Prompting vs. fine-tuning vs. RAG** represent a spectrum of customization, cost, and control; most institutions start with prompting and RAG
- 4 **RAG pipelines** ground LLM outputs in authoritative documents, providing the auditability that finance demands
- 5 **Hallucination is inherent**, not a bug to be fixed — mitigation strategies (grounding, guardrails, human review) are mandatory
- 6 **Token economics** drive deployment costs; model selection is a financial decision, not just a technical one
- 7 **Responsible AI governance** (explainability, fairness, accountability) is a regulatory requirement, not an optional feature
- 8 **Most institutions are early-stage** (experimental or pilot) — the competitive advantage goes to those who move to production with proper governance
- 9 **Emerging frontier: AI agents** — LLMs equipped with tools, memory, and autonomy can execute multi-step financial workflows, moving beyond generation to action

LLMs are transforming financial services, but the institutions that succeed will be those that deploy with appropriate guardrails, not the fastest to adopt.

This lesson:

- Explained the transformer architecture and attention mechanism
- Mapped LLM use cases across financial services
- Compared fine-tuning, prompting, and RAG as deployment strategies
- Designed a RAG pipeline for financial document Q&A
- Evaluated hallucination risks and mitigation strategies
- Analyzed cost-benefit tradeoffs including token economics

Next lesson — Lesson 5.3: Intelligent Automation and RPA:

- How do LLMs integrate with robotic process automation?
- Rule-based automation vs. AI-augmented automation
- Workflow orchestration for financial operations
- Measuring automation ROI in front, middle, and back office

Generative AI provides the intelligence; automation provides the execution. The next lesson connects these into end-to-end workflows.

Attempt these before turning the page.

- 1 [Understand] Distinguish prompt engineering, retrieval-augmented generation (RAG), and fine-tuning. For “answer questions about our internal compliance manual,” which is most appropriate?
- 2 [Apply] GPT-4-class API pricing 2025: \$2.50 / 1M input tokens, \$10.00 / 1M output. Daily query volume 10,000 queries, avg 2,000 input + 500 output tokens. Compute monthly cost.
- 3 [Evaluate] An LLM summarises 10-K filings for analysts. Management changes one word (“increase” to “decrease”) in MD&A, and the LLM summary does not reflect it. Whose responsibility is this: the vendor, the analyst, or the compliance officer? Argue.

Solutions hidden unless `\solutionstrue` is set before compiling.