Three Modalities of Production AI in Financial Services:

Retrieval, Generation, and Detection

A thesis submitted in fulfillment of the requirements for the degree of

Master of Computer Science

at the Visual Intelligence and Learning Laboratory

in collaboration with Bank Lombard Odier & Co

École Polytechnique Fédérale de Lausanne

Peter Abdel Massih

Thesis Advisor:
Prof. Amir Zamir

External Supervisors:
Mickael Brée
Vivien Cuisinier

Lausanne, EPFL, 2026

Acknowledgements

I would like to thank Mickael Brée, Vivien Cuisinier, and Yassine Lajmi for their daily collaboration, technical guidance, and supervision throughout this project.
I would also like to express my sincere gratitude to the wider Content and Communication team at Lombard Odier for their support and valuable discussions.
I am deeply grateful to Prof. Dr. Amir Zamir for giving me the opportunity to carry out my Master's thesis within his lab.

EPFL
Professor Amir Zamir

Lombard Odier
Mickael Brée
Vivien Cuisinier
Yassine Lajmi
Murali Thillaivasan
Simon Richard
Charly Rasclard
Mohammed Bakhti
Jérôme Demanghon
Arnaud Sénéclauze

I sincerely thank my parents for their support throughout difficult times, and my sisters for their constant encouragement. Their assistance and guidance have been indispensable during my time at EPFL.

Lausanne, March 27, 2026 P. A. Massih

Abstract

We present the design, implementation, and production deployment of three AI systems at Lombard Odier, a Swiss private bank, each targeting a distinct operational bottleneck with a different AI modality. The first is a retrieval-augmented generation (RAG) architecture with page-aware chunking that reduces document indexation time by 95.6% and achieves 95% page citation accuracy on a 20-question French-language benchmark, with a 30-variant GGUF quantization ablation showing that imatrix-calibrated K-quants compress embedding models by 70% with negligible quality loss. The second is ALBA, an on-premise LLM advisory pipeline where expert-curated context injection enables a 3B-active-parameter model to outperform GPT-5.1 on domain-specific advisory quality (96% vs. 47% overall), suggesting that context quality matters more than model capability for structured financial reasoning. The third is a regression-based anomaly detection system for tax statements that achieves 80% precision on escalated cases, requiring no labeled anomalies, after reducing the false positive rate from 90% to 5% through domain-informed preprocessing. All three systems are deployed in production under FINMA regulatory constraints and are used by bankers, fund screening operators, and tax specialists. We extract cross-cutting deployment lessons on integrating AI into regulated financial environments, finding that integration engineering consistently exceeds core model development in effort and that task-matched architectures outperform more general alternatives across all three modalities.

Contents

List of Figures

List of Tables

1 Introduction

Every morning, a private banker at Lombard Odier sits down to prepare client meetings. The bank's Chief Investment Officer has just published a new macro outlook recommending reduced exposure to US equities. The equity research team has released flash reports on a dozen stocks. Credit analysts have updated their views on several bond issuers. To give sound advice, the banker needs to synthesize all of this information, often scattered across dozens of internal documents, before the first client call of the day. In practice, this means searching, reading, and cross-referencing reports manually.

The problem extends well beyond the advisory desk. Fund screening operators need to verify whether a newly proposed fund invests in sanctioned jurisdictions or holds excessive cryptocurrency exposure. This verification requires reading fund prospectuses that routinely exceed 600 pages. Meanwhile, tax specialists are reviewing client statements that can contain over 100,000 lines of dividend and position data, cross-referencing them against the Swiss Federal Tax Administration's (AFC) ICTax reference files, which themselves contain millions of entries covering every security traded in Switzerland (Swiss Federal Tax Administration (ESTV/AFC), 2024). A single error in an exchange rate or a withholding amount can cost a client thousands of Swiss francs.

These bottlenecks are not unique to Lombard Odier. According to McKinsey, knowledge workers spend approximately 20% of their time, roughly one full day per working week, searching for and gathering information (McKinsey Global Institute, 2012). In financial services, the problem is amplified by the volume and heterogeneity of internal research produced continuously across departments. The banker who cannot efficiently access this information makes slower decisions. The fund screener who must read 600 pages manually introduces delays. The tax specialist who reviews statements by hand misses errors that automated systems would catch.

Recent advances in large language models (LLMs) and retrieval-augmented generation (RAG) have demonstrated strong capabilities for question answering over document collections (Lewis et al., 2020; Gao et al., 2024). In parallel, LLMs are increasingly explored for

financial decision support, from sentiment analysis (Araci, 2019) to portfolio recommendation (Li et al., 2025; Fieberg et al., 2025). On the compliance side, machine learning methods for anomaly detection in financial data have shown promise, with recent work achieving over 80% precision in fraud detection on tax filings (Vanhoeyveld et al., 2020; De Roux et al., 2018).

However, a significant gap remains between research prototypes and production systems. A recent systematic review of 63 enterprise RAG deployments found that 80.5% still rely on standard retrieval frameworks and that the majority remain in the experimental phase (Karakurt and Akbulut, 2026). In financial services specifically, data sovereignty requirements, on-premise infrastructure constraints, and regulatory obligations under frameworks such as FINMA Guidance 08/2024 (Swiss Financial Market Supervisory Authority (FINMA), 2024) create deployment challenges that academic benchmarks do not capture.

This thesis bridges that gap. We present the design, implementation, and production deployment of three AI systems at Lombard Odier, a Swiss private bank founded in 1796 and managing over CHF 300 billion in client assets. Each system targets a distinct operational bottleneck and employs a different modality of applied intelligence. All three systems are deployed in production and used daily by bankers, tax specialists, and fund screening operators. Beyond the research and experimentation, we managed the full engineering lifecycle, including production releases, CI/CD pipelines, unit testing, and integration with the bank's existing Java microservices and Kafka (Kreps et al., 2011) event infrastructure.

1.1 Three Systems, One Data Ecosystem

The three systems all draw from Lombard Odier's shared data ecosystem but address fundamentally different problems with different AI modalities.

Retrieval-augmented generation for knowledge access. Bankers, investment research teams, and fund screening operators need to query the bank's internal document corpus in natural language and receive answers grounded in specific pages of the source documents. We designed and deployed a production RAG architecture with semantic splitting and page-aware chunking, where each chunk retains its source-page metadata so the LLM can cite specific pages. Through iterative optimization of the embedding inference path, we reduced indexation time for a 52-page fund prospectus from 16 minutes to 44.3 seconds. The system is horizontally scaled on Elasticsearch with durable writes and concurrent multi-user access. Fund screening operators can now ask "Does this fund have exposure to North Korea?" or "What is the crypto allocation?" and receive page-referenced answers from prospectuses that would otherwise require hours of manual reading.

LLM-based advisory for decision support. Translating the bank's macro investment views and individual analyst ratings into concrete trades for a specific client portfolio is time-consuming

and typically done manually. We introduce ALBA (Augmented Layer for Bespoke Advisory), an on-premise LLM advisory pipeline that fuses a client's portfolio with expert-curated summaries of CIO macro outlooks, equity research ratings, and sector views. Rather than relying on automated retrieval, the banker identifies the key documents and we distill each into a structured summary injected into the prompt. This ensures the context is precisely aligned with the advisory logic bankers actually follow. The pipeline operates entirely on-premise under strict data sovereignty constraints and outputs structured buy and sell recommendations with specific instrument suggestions.

Anomaly detection for compliance oversight. Tax statements containing over 100,000 lines of numerical data must be cross-referenced against the AFC's ICTax reference database. Unlike prior work on tax fraud detection that requires labeled examples of fraudulent filings (De Roux et al., 2018; Vanhoeyveld et al., 2020), we propose a regression-based approach using Elastic-Net (Zou and Hastie, 2005) that learns expected column relationships and flags production residuals that deviate from the training distribution, requiring no labeled anomalies. Through iterative refinement of feature engineering and threshold calibration, we reduced the false positive rate from 90% to 5% while maintaining 80% precision on escalated cases. The system is deployed across the bank's Swiss client base.

1.2 Contributions

The main contributions of this thesis are:

1. **A production RAG architecture with semantic splitting and page-aware chunking** (chapter 3). We introduce page-aware chunking where each semantic chunk retains source-page metadata, enabling the LLM to cite specific pages in its responses. The system reduces indexation time for a 52-page PDF from 16 minutes to 44.3 seconds, achieves 95% page citation accuracy on a 20-question French-language benchmark, and is horizontally scaled with durable writes and concurrent access. A 30-variant GGUF quantization ablation demonstrates that imatrix-calibrated K-quants compress embedding models by 70% with negligible quality loss. The system is deployed for fund screening, equity research, and internal document Q&A.

2. **ALBA: an on-premise LLM advisory pipeline for tailored portfolio recommendations** (chapter 4). The pipeline fuses macro investment views, equity research signals, and client portfolio data to generate structured buy and sell recommendations through expert-curated context injection. A 3B-active-parameter on-premise model achieves 96% overall advisory quality versus 47% for GPT-5.1 with web search across three test portfolios, demonstrating that context quality dominates model capability for structured financial reasoning. The pipeline runs entirely on-premise under strict data sovereignty constraints.

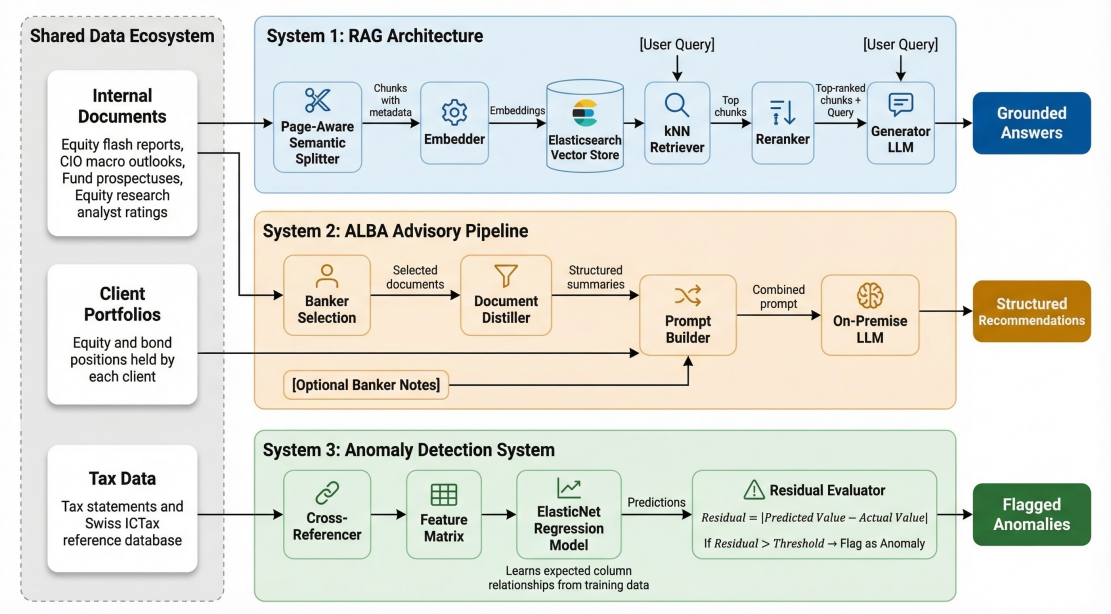

Figure 1.1: Overview of the three systems. Retrieval-augmented generation for knowledge access (Chapter 3), expert-curated LLM advisory for decision support (Chapter 4), and regression-based anomaly detection for compliance oversight (Chapter 5).

3. **A regression-based anomaly detection system for tax statements** (chapter 5). Using ElasticNet with interaction features and grouped cross-validation, the system learns expected column relationships and flags production residuals that deviate from the training distribution, requiring no labeled anomalies. It achieves 80% precision on escalated cases and is deployed across the bank's Swiss client base.

4. **Cross-cutting deployment lessons** (chapter 6) on deploying three distinct AI modalities in a regulated Swiss banking environment, including a comparative analysis of failure modes, regulatory implications, and generalizable insights for the ML systems community.

All three systems are deployed in production, have undergone full testing cycles, and are used by real users at the time of writing.

1.3 Thesis Overview

Figure 1.1 illustrates how the three systems relate to the bank's shared data ecosystem. Chapter 2 reviews the relevant literature. Chapters 3–5 present each system's design, implementation, and experimental evaluation. Chapter 6 synthesizes cross-cutting deployment lessons through the lens of hidden technical debt in ML systems (Sculley et al., 2015) and concludes with limitations and future directions.

2 Background and Related Work

This chapter introduces the foundational concepts and prior work that underpin the three systems presented in this thesis. We cover Transformer-based language models and embeddings, information retrieval and retrieval-augmented generation, regularized regression and anomaly detection, production machine learning challenges, and related work positioning our contributions.

2.1 Transformers and Large Language Models

2.1.1 From Attention to Pre-trained Language Models

The Transformer architecture (Vaswani et al., 2017) replaced recurrent and convolutional sequence models with a mechanism based entirely on attention, removing the sequential bottleneck of recurrent networks and enabling full parallelization during training. The original design follows an encoder-decoder structure: the encoder builds bidirectional representations of the input through stacked layers of multi-head self-attention and feed-forward networks, while the decoder generates output tokens autoregressively with causal attention.

This architecture gave rise to two dominant pre-training paradigms that directly underpin our systems. The first is the encoder-based approach, exemplified by BERT (Devlin et al., 2019), which uses bidirectional masked language modeling to produce contextualized token representations suited for classification, retrieval, and semantic similarity. The second is the decoder-based autoregressive approach, exemplified by the GPT family (Radford et al., 2018; Brown et al., 2020), which predicts the next token given all preceding tokens. This left-to-right generation process is the mechanism behind all LLM text generation, including the grounded answers produced by our RAG system and the structured recommendations generated by ALBA.

A critical capability for our work is in-context learning: large language models can perform tasks from prompt demonstrations alone, without updating model weights (Brown et al.,

2020). Rather than fine-tuning a model on bank-specific data, we provide the relevant context directly in the prompt. This design choice is fundamental to both of our LLM-based systems and is further motivated by evidence that foundation models with in-context learning can match or exceed fine-tuned alternatives on financial tasks (Fieberg et al., 2025), as discussed in Section 2.5.2.

A further scaling strategy is the Mixture-of-Experts (MoE) architecture (Shazeer et al., 2017), which routes each input to a subset of specialized sub-networks so that only a fraction of the total parameters are active per forward pass. We use an MoE model for on-premise generation in Chapter 4.

2.1.2 Embeddings and Representation Learning

Pre-trained language models produce dense vector representations of text (embeddings) where semantically similar texts are close in vector space. These embeddings are the foundation of the retrieval component in our RAG system: documents are split into chunks, each chunk is embedded, and at query time the user's question is embedded using the same model and the nearest chunks are retrieved.

The quality of embeddings varies significantly across models. The Massive Text Embedding Benchmark (MTEB) (Muennighoff et al., 2023) evaluates embedding models across 8 tasks and 58 datasets. A key finding is that no single model dominates all tasks. Model selection must be guided by the specific downstream task. In our system, we began with MXBAI-embed-large, a BERT-based encoder with CLS pooling, before transitioning to alternative models as part of the optimization path described in Chapter 3.

Encoder-Based vs. Decoder-Based Embeddings

How an embedding vector is extracted from a Transformer depends on the model architecture. Three distinct extraction methods exist, illustrated in Figure 2.1.

Encoder-based embeddings. Models such as Sentence-BERT (Reimers and Gurevych, 2019) use bidirectional encoders where every token attends to every other token. Two pooling strategies are common: CLS pooling extracts the hidden state of the `[CLS]` token, while mean pooling averages all token hidden states weighted by the attention mask.

Decoder-based embeddings. A more recent approach uses decoder-only LLMs as the embedding backbone. Models such as Qwen3-Embedding (Zhang et al., 2025) append a special end-of-sequence (EOS) token after the input. Because causal attention ensures each token only attends to preceding tokens, the EOS token at the final position has attended to the entire sequence through accumulated representations across all layers. Its hidden state serves as a compressed summary of the full input, a design known as last-token pooling. This allows embedding models to inherit the long context windows of modern LLMs: Qwen3-Embedding

supports 32,768 tokens versus BERT's 512. In our system, we use Qwen3-Embedding-0.6B for on-premise workloads, which ranks 13th on the MTEB Multilingual v2 leaderboard as of March 2026 despite being the smallest model in the top 15 (Zhang et al., 2025). Our embedding optimization path in Chapter 3 traverses both methods.

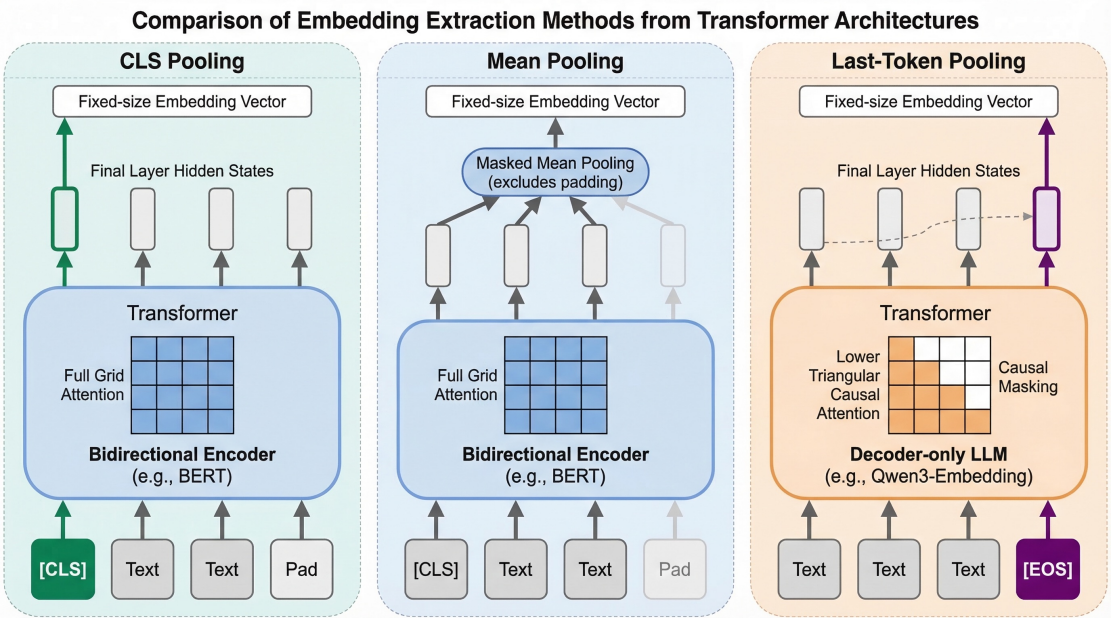

Figure 2.1: Three approaches for extracting text embeddings. CLS pooling (left) uses the `[CLS]` token from a bidirectional encoder. Mean pooling (center) averages all token hidden states. Last-token pooling (right) uses the final `[EOS]` token from a decoder with causal attention.

In practice, the pooling strategy is a hard requirement tied to the model architecture. Inference frameworks such as llama.cpp (Gerganov, 2023) expose this as a `-pooling` parameter: `cls` for models such as MXBAI-embed-large and BGE-M3 (Chen et al., 2024), `mean` for models such as all-MiniLM-L6-v2, `last` for decoder-based models like Qwen3-Embedding, and `rank` for reranking models (Section 2.2.5). Using the wrong mode might produce poor-quality embeddings.

2.2 Information Retrieval and Retrieval-Augmented Generation

2.2.1 Sparse and Dense Retrieval

Traditional retrieval uses sparse representations based on term frequency. BM25 (Robertson and Zaragoza, 2009) scores documents by term overlap with the query, weighted by inverse document frequency. It is fast and requires no training data but cannot capture semantic similarity: a query about "stock market decline" will not match a document discussing "equity selloff."

Dense retrieval addresses this by representing both queries and documents as dense vectors

in a shared embedding space. Karpukhin et al. (2020) introduced Dense Passage Retrieval (DPR) using dual BERT encoders trained on question-passage pairs, demonstrating 9 to 19 percentage points higher top-20 accuracy over BM25.

2.2.2 Approximate Nearest Neighbor Search with HNSW

Dense retrieval requires efficient nearest neighbor search over potentially millions of vectors. The Hierarchical Navigable Small World (HNSW) algorithm (Malkov and Yashunin, 2020) builds a multi-layer graph structure (Figure 2.2) where upper layers contain few nodes with long-range connections for coarse navigation and lower layers are progressively denser. Search proceeds greedily from the top layer and descends, yielding $O(\log n)$ complexity.

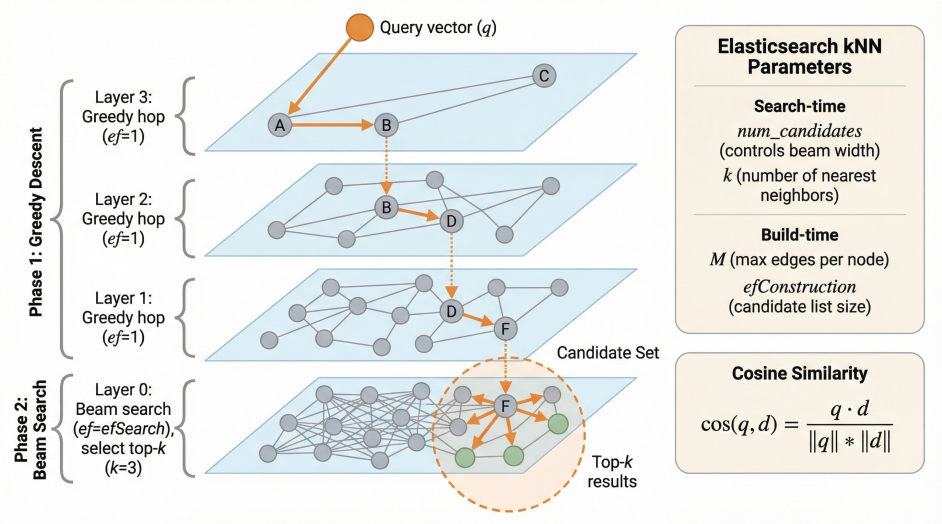

Figure 2.2: The HNSW graph structure. Upper layers contain fewer nodes with long-range connections for coarse navigation. Lower layers are progressively denser. Search proceeds in $O(\log n)$ complexity. The position of q is schematic, indicating the query entry point.

Elasticsearch (Elastic NV, 2024) implements HNSW as its default vector search algorithm since version 8.0, supporting cosine similarity, dot product, and L2 distance. Our RAG system uses Elasticsearch for all document embeddings, combining HNSW search with durable writes and concurrent access. The specific configuration is discussed in Chapter 3.

2.2.3 The RAG Paradigm

Retrieval-Augmented Generation (RAG), introduced by Lewis et al. (2020), combines a retriever with a generator. Instead of relying solely on the knowledge stored in an LLM's parameters, the model retrieves relevant documents at inference time and conditions its generation on them. This addresses two fundamental limitations of standalone LLMs: they cannot access information beyond their training data, and they are prone to hallucination.

Gao et al. (2024) categorize RAG systems into three generations: Naive RAG (simple retrieve-then-read), Advanced RAG (with query optimization, post-retrieval reranking, and iterative refinement), and Modular RAG (with interchangeable components). Our system falls into the Advanced RAG category, incorporating semantic chunking, page-aware metadata, and reranking.

2.2.4 Document Chunking Strategies

Before documents can be indexed in a vector store, they must be split into chunks that are small enough to embed meaningfully but large enough to retain context. The simplest approach is fixed-size chunking, which splits documents into segments of a fixed token count with optional overlap. This is deterministic but ignores document structure. Recursive character splitting improves on this by splitting at natural boundaries (paragraphs, sentences) in hierarchical order. Empirical evaluations suggest chunks of 400 to 512 tokens with 10 to 20% overlap provide a good default (Bhat et al., 2025).

Semantic chunking identifies natural topic boundaries by measuring cosine similarity between successive sentence embeddings. When the similarity drops below a threshold, a new chunk begins. This produces variable-length chunks aligned with the document's semantic structure.

Our system uses page-aware chunking, where each chunk retains metadata about the source pages it spans. This enables the LLM to cite specific pages in its responses, a requirement for document-grounded question answering in a banking context where traceability matters. The technical details are presented in Chapter 3.

2.2.5 Reranking

The initial retrieval step in a RAG pipeline uses bi-encoders that embed queries and documents independently, enabling fast nearest-neighbor search but limiting expressiveness since no interaction occurs between query and document representations. Reranking recovers this precision by re-scoring a candidate set. Three architectures exist, illustrated in Figure 2.3.

Cross-encoder reranking. The query and document are processed jointly as a single input through a bidirectional encoder, allowing full attention between all tokens. A classification head produces a single relevance score. This captures fine-grained interactions that bi-encoders miss but requires a full forward pass for every query-document pair.

Generative reranking. Decoder-based rerankers such as Qwen3-Reranker (Zhang et al., 2025) frame relevance assessment as a binary next-token decision. Given an instruction I, a query q, and a document d, the model computes the logits for the next token being "yes" or "no" and converts them into a relevance score via a two-way softmax:

$$\text{score}(q,d) = \frac{e^{z_{\text{yes}}(I,q,d)}}{e^{z_{\text{yes}}(I,q,d)} + e^{z_{\text{no}}(I,q,d)}} \tag{2.1}$$

where z_{yes} and z_{no} are the next-token logits conditioned on the instruction, query, and document. This formulation achieves strong reranking performance while benefiting from long-context support. We use Qwen3-Reranker as a multilingual reranker in Chapter 3.

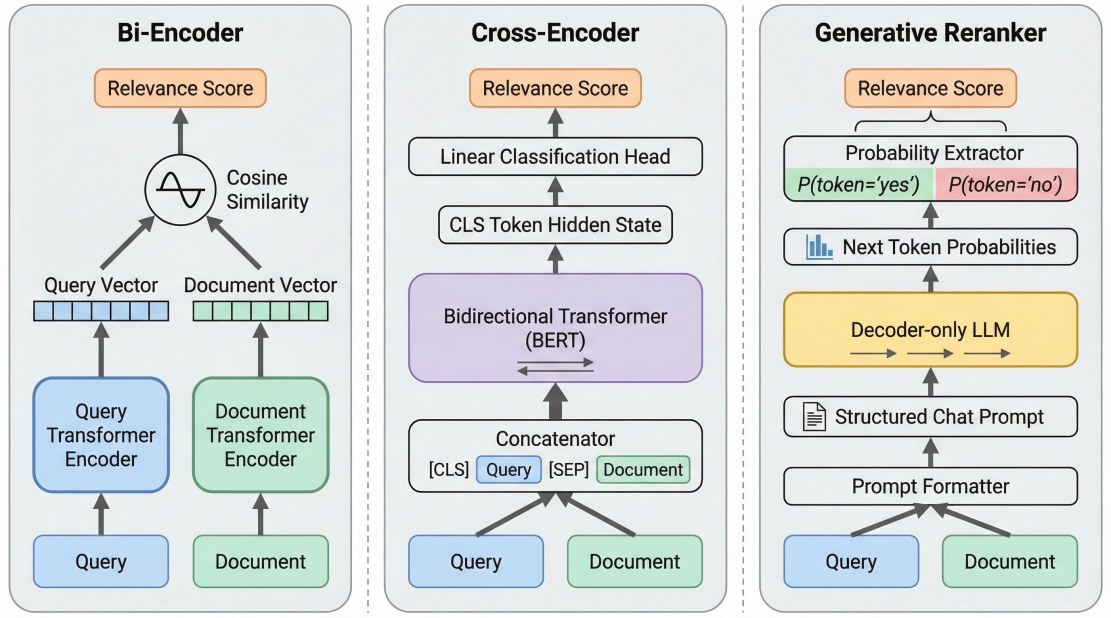

Figure 2.3: Three architectures for retrieval and reranking. Bi-encoders (left) encode independently for fast retrieval. Cross-encoders (center) process the pair jointly for accurate reranking. Generative rerankers (right) compute relevance via next-token logits for "yes" vs "no" from a decoder-only LLM (Equation 2.1).

In practice, the two-stage retrieve-then-rerank pattern combines a bi-encoder for fast initial retrieval with a reranker for precise re-scoring. This is the strategy we use in our RAG system.

2.3 Regularized Regression and Anomaly Detection

2.3.1 ElasticNet Regularization

When predictor variables are highly correlated, as in tax statement data where columns such as gross dividend, withholding amount, and net amount are arithmetically related, ordinary least squares regression produces unstable estimates. Regularization addresses this by adding penalty terms. Ridge regression (Hoerl and Kennard, 1970) adds an L2 penalty that shrinks coefficients proportionally but never eliminates them. The Lasso (Tibshirani, 1996) uses an L1 penalty that drives some coefficients exactly to zero, performing variable selection, but arbitrarily selects among correlated predictors.

ElasticNet (Zou and Hastie, 2005) combines both:

$$\hat{\beta}^{\text{elastic}} = \arg\min_{\beta} \sum_{i=1}^{n} (y_i - x_i^\top \beta)^2 + \lambda \left[\alpha \sum_{j=1}^{p} |\beta_j| + (1-\alpha) \sum_{j=1}^{p} \beta_j^2 \right] \tag{2.2}$$

where $\alpha \in [0,1]$ balances L1 and L2 regularization and λ controls overall penalty strength. The key advantage is the grouping effect: when predictors are correlated, ElasticNet selects or excludes them together rather than arbitrarily picking one. This is relevant to our tax data, where the feature matrix exhibits both high multicollinearity and significant sparsity (many fields are absent for certain security types). The L2 component handles correlated columns while L1 performs automatic feature selection. The constraint regions for all three methods are illustrated in Figure 2.4.

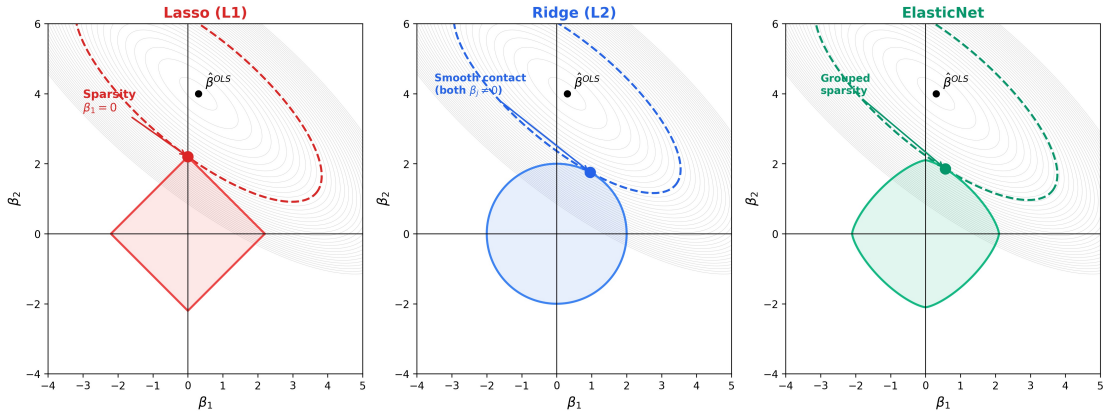

Figure 2.4: Constraint regions for Lasso (left, diamond), Ridge (center, circle), and ElasticNet (right, rounded diamond) in two-dimensional coefficient space. The grey contours represent the OLS loss surface. The diamond corners induce sparsity by forcing coefficients to zero.

2.3.2 Anomaly Detection via Residual Analysis

Anomaly detection identifies observations that deviate significantly from expected behavior (Chandola et al., 2009). Our method is statistical anomaly detection using regression: we train a model to predict a target column from the remaining columns and flag observations where the residual $r_i = |y_i - \hat{y}_i|$ exceeds a threshold calibrated from the training distribution.

This approach has a key advantage over classification-based methods: it requires no labeled examples of anomalies. In our setting, the bank does not maintain a database of known data-entry errors. The regression approach learns what normal data looks like and flags deviations, making it a form of unsupervised anomaly detection.

2.3.3 Cross-Validation for Financial Data

Standard k-fold cross-validation assumes independent observations. In tax data, records from the same client share filing patterns and jurisdictional rules. Grouped cross-validation (Pedregosa et al., 2011) assigns all records from the same client to the same fold, preventing the model from memorizing client-specific patterns. Our configuration is detailed in Chapter 5.

2.4 Production Machine Learning Systems

2.4.1 Technical Debt in Machine Learning

Sculley et al. (2015) introduced the concept of hidden technical debt in ML systems, arguing that the ML code constitutes only a small fraction of the overall system. The surrounding infrastructure, data collection, feature extraction, configuration management, serving, and monitoring, dwarfs the modeling component. They identified several patterns directly relevant to our deployments: *glue code* arising from Python/Java interoperability, *pipeline jungles* from accumulating preprocessing steps (particularly in our tax anomaly system), and *configuration debt* from hyperparameters and environment-specific settings across development, staging, and production.

2.4.2 AI Regulation in Swiss Financial Services

FINMA published Guidance 08/2024 on AI governance and risk management in December 2024 (Swiss Financial Market Supervisory Authority (FINMA), 2024), establishing expectations across seven areas: governance, AI inventory with risk classification, data quality, testing and validation, documentation, explainability, and independent review. The guidance applies existing principle-based regulation to AI rather than prescribing technical requirements.

For our deployments, this has concrete implications. The RAG system must provide traceable, page-referenced answers. The ALBA pipeline must be explainable, which is why we inject curated document summaries rather than opaque embedding-based context. The anomaly detection system must have documented and justifiable thresholds. Switzerland does not have AI-specific legislation; FINMA's approach contrasts with the EU AI Act (Regulation 2024/1689), which classifies credit scoring as high-risk AI. Swiss banks serving EU clients face dual regulatory considerations.

2.4.3 Model Optimization and Deployment

Deploying ML models in production requires optimizing inference beyond research frameworks. ONNX Runtime (ONNX Runtime developers, 2021) provides a hardware-agnostic engine with graph-level optimizations (operator fusion, constant folding, memory planning), achieving significant speedups over native PyTorch on CPU hardware. For more constrained

environments, llama.cpp (Gerganov, 2023) enables LLM inference in pure C/C++ using the GGUF file format, which encodes quantized weights, tokenizer, and metadata in a single binary optimized for CPU inference via SIMD instructions.

Quantization reduces model weights to lower precision (2–8 bits), trading output quality for reduced memory and latency. GGUF supports three quantization families: *legacy types* (Q4_0 through Q8_0) with uniform round-to-nearest quantization per 32-weight block; *K-quants* (Q2_K through Q6_K) with hierarchical super-blocks and mixed-precision layer assignment (_S, _M, _L variants); and *I-quants* (IQ1_S through IQ4_NL) using lattice-based vector quantization with non-linear reconstruction. All three families can use an importance matrix (imatrix), which records the activation magnitudes of each weight during inference on a calibration corpus, allowing the quantizer to preserve precision for the most impactful weights. Table 2.1 summarizes representative levels; we evaluate the full set on financial document embeddings in Chapter 3. Alternative methods such as GPTQ (Frantar et al., 2023) and AWQ (Lin et al., 2024) require GPU hardware, making them unsuitable for our on-premise deployment.

Table 2.1: Representative GGUF quantization levels for a 7B-parameter model.

Type	Family	Bits/weight	Size (7B)
Q8_0	Legacy	8.5	7.0 GB
Q6_K	K-quant	6.6	5.5 GB
Q5_K_M	K-quant (mixed)	5.7	4.8 GB
Q4_K_M	K-quant (mixed)	4.8	3.8 GB
IQ4_NL	I-quant	4.5	3.6 GB
IQ3_S	I-quant	3.5	3.0 GB
Q2_K	K-quant	2.6	2.7 GB

2.5 Related Work

2.5.1 RAG Systems in Enterprise Settings

Karakurt and Akbulut (2026) surveyed 63 enterprise RAG deployments and found that 80.5% rely on standard retrieval frameworks, with the majority remaining experimental. None of the surveyed systems report page-level citation. Most RAG evaluations use Wikipedia-based benchmarks where documents are short and well-structured. Financial documents are different: fund prospectuses span hundreds of pages, equity research reports mix narrative with quantitative tables, and macro investment views are updated daily. Unlike prior work, we deploy and evaluate a RAG system on heterogeneous financial documents in a production banking environment with strict latency and traceability requirements.

2.5.2 LLMs for Financial Advisory

LLM applications in finance have progressed along two paths. BloombergGPT (Wu et al., 2023), a 50B-parameter model trained on 700B tokens with 1.3M GPU hours, represents the domain-specific approach but is impractical for most institutions. FinGPT (Yang et al., 2023) and FinBERT (Araci, 2019) offer lighter alternatives through fine-tuning.

For advisory tasks specifically, Fieberg et al. (2025) tested 32 LLMs across 64 investor profiles and found that foundation models outperform fine-tuned models, supporting our design choice: rather than training a financial LLM, ALBA uses a general-purpose model with expert-curated context. MarketSenseAI (Fatouros et al., 2025) combines GPT-4 with chain-of-thought prompting for stock recommendations but relies on cloud API calls. ALBA operates entirely on-premise. While Li et al. (2025) combines LLMs with reinforcement learning for portfolio recommendations, our design prioritizes production viability through structured prompting over learned policies.

2.5.3 Anomaly Detection in Tax and Financial Data

De Roux et al. (2018) addressed under-reporting in Colombian tax declarations using unsupervised ML at KDD 2018, motivated by the absence of labeled audit data. Vanhoeyveld et al. (2020) proposed a multi-module framework for VAT fraud detection in Belgium. Both target intentional fraud. Our setting is fundamentally different: we detect accidental data-entry errors in statements produced by the bank's own systems, with no labeled anomaly dataset. This drives our methodological choice of regression-based residual analysis over supervised classification.

3 Scalable RAG Architecture for Financial Document Retrieval

3.1 Problem Formulation

The bank's internal document corpus, introduced in Chapter 1, spans equity flash reports, macro investment proposals, research lists, and fund prospectuses. Users need answers grounded in specific pages: a compliance officer verifying a fund screening decision must know that the answer came from page 47 of the prospectus, not from an unattributed chunk.

Indexing a single 52-page fund prospectus initially took approximately 16 minutes, and generating the first response token required several minutes of retrieval and context assembly. The system supported a single user, had no persistence guarantees, and could not scale beyond a handful of documents.

We set out to build a production RAG system with four requirements: (1) sub-minute indexation to support continuous ingestion, (2) page-level citation accuracy, (3) horizontal scaling with durable writes for concurrent multi-pod access, and (4) interactive query latency. This chapter presents the system design, describes the iterative embedding optimization path from HuggingFace through ONNX Runtime to llama.cpp and finally Azure-hosted models, and evaluates the system across three experiments: a quantization ablation over 30 GGUF variants, an indexation time progression across four inference frameworks, and a retrieval quality comparison across chunking and reranking configurations.

3.2 System Design

Figure 3.1 illustrates the end-to-end pipeline. A document arrives at the Java backend, which forwards it to the Python Flask service. The service extracts text (page by page for PDFs, pre-extracted for other formats), summarizes the document, splits it into semantic chunks with page metadata when available, and embeds each chunk. The embedded chunks are POSTed back to the Java backend via an HTTP callback, and the Java backend writes them into Elasticsearch. At query time, the user's question is contextualized, embedded, and used to

retrieve the top 300 chunks via kNN (k-nearest neighbors) search. When the retrieved context exceeds the generation model's token limit, the chunks are reranked and pruned to the largest subset that fits. The selected chunks are assembled into a structured XML prompt and passed to GPT-5.1, which generates a response with page-level citations.

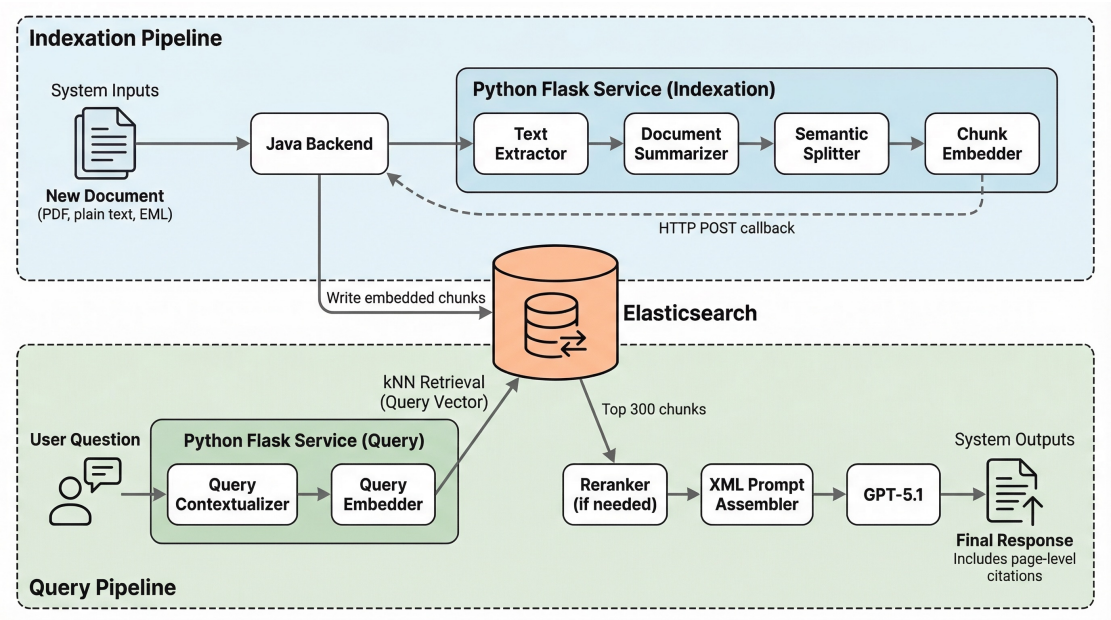

Figure 3.1: End-to-end RAG pipeline. Documents (PDF, plain text, EML...) are processed, summarized, split into semantic chunks, embedded, and ingested into Elasticsearch via a Java callback. At query time, chunks are retrieved via kNN search, optionally reranked and pruned (Algorithm 1), assembled into an XML prompt, and passed to GPT-5.1 for generation with page-level citations.

3.2.1 Document Processing Pipeline

The system accepts documents through two ingestion paths. For PDFs, the file is uploaded via multipart form data and extracted page by page using pdfplumber, producing a separate text block per page tagged with the source page number. For other formats (plain text, EML, and similar text-based documents), the text arrives pre-extracted via a JSON API with a unique reference identifier, title, and creation date. The `page` field is optional and typically absent for non-paginated formats. Pdfplumber handles standard text extraction but does not interpret tables or charts embedded as images.

Text preprocessing applies two domain-specific transformations. First, regulatory disclaimers appended to investment research documents are removed. These disclaimers, which appear in both English ("IMPORTANT INFORMATION") and French ("INFORMATION IMPORTANTE"), can span up to 4,000 characters and would otherwise pollute the semantic content of terminal chunks. The removal is conservative: it only triggers when the disclaimer appears in the final 4,000 characters of the document, avoiding false positives on documents that discuss

regulatory topics in their body text. Second, redundant consecutive newlines are collapsed to single newlines to normalize whitespace across documents from different PDF generators.

3.2.2 Semantic Splitting with Page-Aware Chunking

We use the SemanticSplitterNodeParser from LlamaIndex (LlamaIndex, 2024) with a buffer size of 8 sentences and a breakpoint percentile threshold of 99. The buffer size controls how many sentences are grouped before evaluating whether a semantic boundary has been crossed. At the 99th percentile, only the most significant topic transitions produce chunk boundaries, resulting in longer, more coherent chunks that preserve the narrative structure of financial documents. With these settings, the splitter produces chunks of up to 989 tokens, typically 1–2 chunks per page.

The critical design choice is that each page is processed as a separate Document object, with the page number stored in the metadata. When the semantic splitter produces chunks from a page, each chunk inherits the page metadata from its parent. Every chunk in the vector store therefore carries a `page` field identifying the source page, enabling the LLM to cite specific pages in its responses. The splitting operates within individual pages rather than across page boundaries, so page attribution is always unambiguous.

For non-PDF documents ingested without page numbers, chunks carry a `0` page field. The XML prompt (Figure 3.2) omits the `number` attribute from the corresponding `<Page>` tag, and the generation instructions direct the LLM to report that page information is unavailable.

In practice, the trade-off is favorable. Most topic transitions in financial documents align with page boundaries, and the cases where a semantic unit spans two pages are rare enough that the loss in chunking quality is negligible compared to the gain in citation accuracy.

Document summarization. Each document is summarized through an iterative process before indexing. The text is split into 20,000-token windows with 1,000-token overlap. For each window, the LLM is prompted with the partial summary accumulated so far and the new text, producing a progressive summary that captures the document's key themes. This summary is attached as metadata to every chunk from that document and stored alongside the embedding in Elasticsearch. At query time, the summary gives the LLM document-level context that helps it synthesize answers from multiple chunks of the same document. Documents shorter than 100 characters are not summarized.

3.2.3 Embedding Optimization Path

The embedding step dominates indexation time because every chunk requires a forward pass through a neural network. We iteratively optimized this bottleneck across four inference frameworks. All timing measurements are averages over 10 runs on the same 52-page fund

prospectus (the Harvest fund), using the same chunking configuration.

Stage 1: HuggingFace (baseline). The initial implementation used MXBAI-embed-large (335M parameters, 512-token context, CLS pooling) loaded via the HuggingFace Transformers library on CPU with default PyTorch settings. This indexed the 52-page document in 1,012 seconds (16 minutes 52 seconds).

Stage 2: ONNX Runtime. We exported the same model to the ONNX intermediate representation and ran inference through ONNX Runtime, which applies graph-level optimizations including operator fusion, constant folding, and memory planning (Section 2.4.3). The model, tokenizer, and inference loop remained identical; only the runtime changed. This reduced indexation time to 512 seconds (8 minutes 32 seconds), a 49% improvement.

Stage 3: llama.cpp. We transitioned from MXBAI-embed-large to Qwen3-Embedding-0.6B (Zhang et al., 2025), a decoder-based embedding model running via llama.cpp (Gerganov, 2023) in a separate sidecar container. This offered three advantages. First, llama.cpp serves embeddings over an HTTP API (`/v1/embeddings`), decoupling the embedding workload from the Python application and allowing independent scaling. Second, the GGUF quantization format (Section 2.4.3) reduces memory footprint while maintaining embedding quality, as we show in Section 3.3.1. Third, llama.cpp supports concurrent request handling (`-parallel 2`), allowing multiple indexation jobs to share the same model instance.

The production configuration uses Q3_K_M-imat quantization (331 MB, 4.66 bits per weight) with 8 threads matching the pod's CPU core count, quantized KV cache (`-cache-type-k q4_1 -cache-type-v q4_1`), batch sizes of 8,192 tokens, and memory locking (`-mlock`) to prevent page faults. This indexed the 52-page document in 232 seconds (3 minutes 52 seconds), a 77% improvement over the baseline.

Stage 4: Azure text-embedding-3-large (production). The current production deployment uses Azure-hosted text-embedding-3-large, accessed via the OpenAI-compatible API through LlamaIndex's OpenAIEmbedding client. This model offers 3,072-dimensional embeddings (versus 1,024 for Qwen3-Embedding-0.6B) and runs on GPU-accelerated infrastructure. Combined with batch embedding (300 texts per request) and parallelized node building (8 workers via ThreadPoolExecutor), this configuration achieves an average indexation time of 44.3 seconds for the 52-page document.

The llama.cpp sidecar container remains deployed alongside the Azure embedding path. It serves as the inference engine for on-premise chat workloads (including ALBA, Chapter 4), and can be reconfigured at runtime by writing new command-line arguments to a shared file and triggering a reload via a watchdog process. This dual-path architecture gives us both the throughput of cloud-hosted embeddings for high-volume indexation and the data sovereignty

of on-premise inference for sensitive workloads.

3.2.4 Reranking

When the full set of $k = 300$ retrieved chunks exceeds the generation model's context limit, the system reranks them before pruning. We use Qwen3-Reranker-0.6B (Zhang et al., 2025), a multilingual generative reranker (Section 2.2.5), loaded via SentenceTransformerRerank from LlamaIndex with a candidate window of 300 chunks. The reranker scores each query-chunk pair and reorders the candidates so that the most relevant chunks appear first.

We chose 300 as the reranking window to balance precision and latency. Financial documents produce many chunks (a 600-page prospectus can yield over 1,000), and the initial HNSW retrieval may return many similar results.

3.2.5 Query Pipeline and Generation

During retrieval, the user's question is first contextualized against the conversation history using a dedicated LLM call that resolves pronouns and references from prior turns (e.g., "What about page 12?" becomes "What does page 12 of the Harvest fund prospectus say about cryptocurrency allocation?"). The contextualized query is then embedded using the same model used for indexation, and the top $k = 300$ nearest chunks are retrieved from Elasticsearch via approximate kNN search, filtered to the allowed document references. If the assembled prompt fits within the context limit, these chunks are used directly. Otherwise, they are reranked (Section 3.2.4) and pruned via Algorithm 1. Table 3.1 shows the schema of each stored chunk.

Table 3.1: Schema of an embedded chunk in Elasticsearch. The `vector` field is indexed for HNSW search; all other fields are returned as metadata at query time.

Field	Type	Description
`id`	string	Unique chunk identifier (node ID or `page_id::cN`)
`reference`	string	Root document identifier (for filtering)
`page_id`	string	Page-level identifier (`reference::pN`)
`page`	integer	Source page number
`text`	string	Chunk text (what gets embedded)
`vector`	float[]	Embedding vector (1,024 or 3,072 dimensions)
`title`	string	Document title
`date`	string	Document creation date
`summary`	string	Document-level iterative summary

Context assembly. The retrieved chunks are assembled into the structured XML prompt shown in Figure 3.2. Chunks are grouped by root document reference, then by page number in ascending order. Duplicate chunk texts within the same page are deduplicated. The result

is a hierarchical structure where each <Document> contains a title, date, summary, and a list of <Page number="N"> elements, preserving document and page boundaries for precise citation.

```
<Prompt>
  <Query>{query}</Query>
  <Documents>
    <Document index="1">
      <Title>...</Title>  <Date>...</Date>
      <Summary>...</Summary>
      <Pages>
        <Page number="12">{chunk texts}</Page>
        <Page number="34">{chunk texts}</Page>
      </Pages>
    </Document>
  </Documents>
  <Instructions>
    Cite using [:doc_index]. For pages: "Fact [:1] Page X"
    using <Page number=NUM> from metadata.
  </Instructions>
</Prompt>
```

Figure 3.2: Structured XML prompt template. Retrieved chunks are organized by document and page. The LLM cites documents via anchor notation and reports page numbers from chunk metadata.

Context window management. The generation model (GPT-5.1, 272,000-token input limit) can be exceeded when retrieving 300 chunks from large documents, particularly with multi-turn conversation history. The system first attempts the full retrieval. Only when it exceeds the limit does it invoke Algorithm 1, a binary search over the ranked result list. Because the token count of the assembled prompt is monotonically non-decreasing with the number of included chunks, binary search is valid and converges in $O(\log k)$ iterations. The chunks are reranked once before the search begins, so pruning always removes the least relevant chunks first. Token counting uses tiktoken with the `o200k_base` encoding, matching GPT-5.1's tokenizer.

Generation. The generation model in production is GPT-5.1, hosted on Azure and accessed via the OpenAI-compatible chat completions API with a 272,000-token input window and 128,000-token output window. The system maintains a conversation memory that accumulates user queries and assistant responses, enabling multi-turn interactions over the same document set.

Algorithm 1: Context window fitting with early exit. Step 1 checks whether the full retrieval fits. If so, no pruning occurs. Step 2 reranks and binary-searches for the largest prefix within the limit, converging in at most $\lceil \log_2 k \rceil$ iterations (8–9 for $k = 300$).

Input: *query*: user query string
nodes: list of k retrieved chunks from Elasticsearch
history_tokens: token count of conversation history
L: effective context limit (with safety margin)
Output: *best_nodes*: largest prefix of ranked nodes fitting within L

```
// Step 1: attempt full retrieval
prompt ← AssembleXmlPrompt(nodes, query);
if history_tokens + CountTokens(prompt) ≤ L then
    return nodes;                                  // fits; no pruning needed
end
// Step 2: rerank once, then binary search
ranked ← Rerank(nodes, query);
lo ← 1;  hi ← k;  best_k ← 0;
while lo ≤ hi do
    mid ← ⌊(lo + hi)/2⌋;
    prompt ← AssembleXmlPrompt(ranked[1..mid], query);
    t ← history_tokens + CountTokens(prompt);
    if t ≤ L then
        best_k ← mid;
        lo ← mid + 1;
    else
        hi ← mid − 1;
    end
end
return ranked[1.. max(1, best_k)];
```

3.2.6 Horizontal Scaling and Java/Python Integration

The initial implementation used LlamaIndex's built-in local vector store, which stores embeddings in memory and persists them to disk. This was sufficient for single-user prototyping but introduced two problems at scale: concurrent writes from multiple pods could corrupt the index, and retrieval latency for 300 chunks was approximately 40 seconds. We migrated to Elasticsearch with HNSW indexing, hosted on the bank's on-premise infrastructure. The same 300-chunk kNN retrieval with document-level filtering now completes in under 10 seconds.

Callback architecture. The system is deployed as a Kubernetes StatefulSet with two replicas, each running three containers: the Python Flask application, the llama.cpp server, and an administration server. All replicas share a 100 GB persistent volume claim (ReadWriteMany) for model weights and runtime state.

The Python codebase cannot write directly to Elasticsearch. Provisioning direct database access for a new codebase at the bank requires security review, technical user provisioning, and network policy changes—a process that typically takes months. We solved this through a callback architecture: the Python application serializes each embedded chunk as a JSON payload via the `EmbeddedChunk` dataclass and POSTs it to a Java endpoint (`/embedding-callback`) within the bank's existing Spring Boot microservices, which handles the actual Elasticsearch write operations.

Each chunk's unique `id` field prevents duplicate writes when multiple pods index the same document concurrently. The retry logic uses exponential backoff with jitter (base 1.5 seconds, maximum 2 hours per retry, 16 attempts) to handle transient failures from rate limiting, network interruptions, and model endpoint unavailability. A bounded semaphore limits concurrent outbound model API calls to 2, reducing rate-limit spikes when multiple background indexation jobs run simultaneously.

3.3 Experimental Evaluation

We evaluate the system through three experiments. The first measures the effect of GGUF quantization on embedding model quality. The second tracks indexation time across the four inference frameworks. The third compares retrieval quality across chunking and reranking configurations.

All experiments use the same benchmark document: a 52-page fund prospectus (the Harvest fund) representative of the fund screening use case. Timing experiments report the average over 10 runs. The hardware configuration for on-premise experiments matches the production deployment: 8 CPU cores, 38 GB memory for the Python container, and 8 CPU cores, 20 GB memory for the llama.cpp container.

3.3.1 Experiment 1: GGUF Quantization Ablation

To determine how aggressively the embedding model can be quantized without degrading retrieval quality, we quantized Qwen3-Embedding-0.6B into 30 GGUF variants spanning all three quantization families described in Section 2.4.3: legacy types, K-quants, and I-quants. For K-quant and I-quant variants, we computed an importance matrix (imatrix) calibrated on a 22 MB mixed-domain corpus weighted toward financial text, including FiQA, FinanceBench SEC 10-K filings, financial sentiment data, and financial RAG pairs, alongside mathematical reasoning and general text from WikiText-2. All quantized models are publicly available.[1]

We measured perplexity using `llama-perplexity` on the same calibration corpus with a context window of 1,536 tokens and 200 chunks. Table 3.2 reports selected results spanning the full bit-width range; the complete 30-variant table is in Appendix B. The BF16 (unquantized) baseline achieves a perplexity of 406.03.

Table 3.2: Selected GGUF quantization results for Qwen3-Embedding-0.6B. Δ PPL is relative to BF16 baseline (406.03). Full 30-variant table in Appendix B.

Model	Family	BPW	Size	Δ PPL
BF16 (baseline)	–	16.08	1.1 GB	–
Q8_0	Legacy	8.58	610 MB	+3.54
Q6_K	K-quant	6.65	472 MB	+11.35
Q5_0	Legacy	5.86	416 MB	+7.17
Q4_K_M-imat	K-quant	5.32	378 MB	+0.65
Q4_K_S-imat	K-quant	5.14	365 MB	+0.97
IQ4_NL-imat	I-quant	5.12	364 MB	+29.00
Q3_K_L-imat	K-quant	4.94	351 MB	+6.00
Q3_K_M-imat	**K-quant**	**4.66**	**331 MB**	**+0.62**
IQ3_M-imat	I-quant	4.51	320 MB	+54.92
Q2_K-imat	K-quant	3.97	282 MB	+391.83
IQ2_M-imat	I-quant	3.55	252 MB	+877.42
TQ2_0-imat	Ternary	3.32	236 MB	diverged
IQ1_S-imat	I-quant	2.79	198 MB	+23008.92

Figure 3.3 plots bits per weight against perplexity delta for all 30 variants. Three regimes are visible. Above 5 BPW, perplexity increases are modest (under +40) but models remain large (364–610 MB). Between 4 and 5 BPW, imatrix-calibrated K-quants achieve near-baseline quality: Q3_K_M-imat at 4.66 BPW and 331 MB shows only +0.62 perplexity over BF16, a 70% size reduction with negligible quality loss. Below 4 BPW, perplexity degrades steeply, eventually diverging for ternary quantizations.

A notable finding is that imatrix-calibrated K-quants outperform I-quants at comparable bit widths. IQ4_NL-imat (5.12 BPW, +29.00) and IQ3_M-imat (4.51 BPW, +54.92) show substan-

[1] https://huggingface.co/PeterAM4/Qwen3-Embedding-0.6B-GGUF

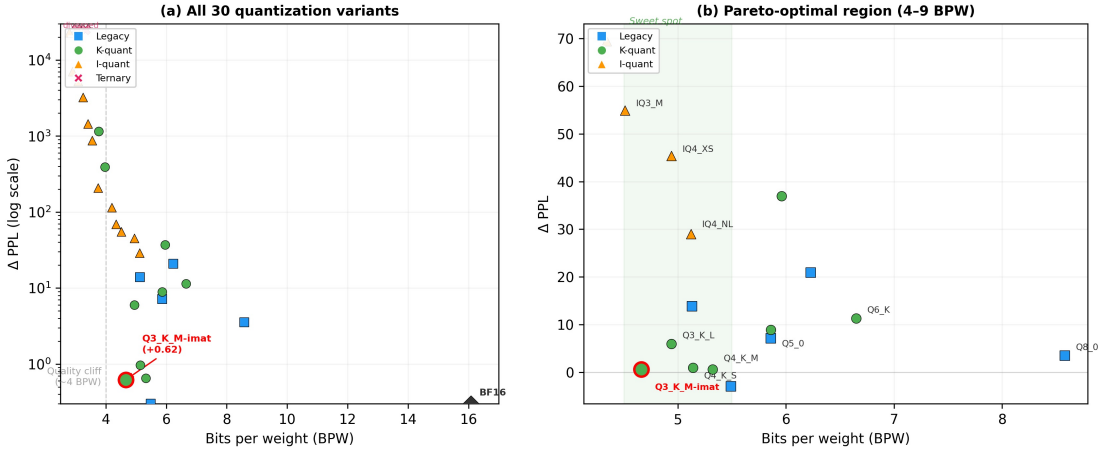

Figure 3.3: Bits per weight versus perplexity delta across all 30 GGUF quantization variants of Qwen3-Embedding-0.6B. (a) Full range on a log scale showing the quality cliff below 4 BPW. (b) Zoomed into the Pareto-optimal region (4–9 BPW). Q3_K_M-imat (331 MB, +0.62 PPL) is the optimal quality-size trade-off.

tially worse perplexity than Q4_K_S-imat (5.14 BPW, +0.97) and Q3_K_M-imat (4.66 BPW, +0.62). This suggests that for small embedding models (0.6B parameters), simpler block-wise K-quant methods with imatrix calibration achieve better results than the lattice-based vector quantization used by I-quants. Based on this analysis, we selected Q3_K_M-imat for our on-premise deployment.

3.3.2 Experiment 2: Indexation Time Progression

Table 3.3 reports the average indexation time across the four inference frameworks. All configurations use the same document processing pipeline and batch embedding (256 chunks per batch for on-premise, 300 for Azure). The timing includes the full pipeline: PDF extraction, preprocessing, semantic splitting, summarization, embedding, and callback to Java.

Table 3.3: Indexation time for a 52-page fund prospectus across four embedding configurations. All values are averages over 10 runs.

Stage	Embedding model	Context	Time (s)	vs. Baseline
1. HuggingFace	MXBAI-embed-large	512	1012	–
2. ONNX Runtime	MXBAI-embed-large	512	512	−49%
3. llama.cpp	Qwen3-Embed-0.6B (Q3_K_M)	32K	232	−77%
4. Azure	text-embedding-3-large	8K	44.3	−95.6%

The progression from 1,012 seconds to 44.3 seconds represents a 95.6% reduction. The largest single improvement came from the HuggingFace-to-ONNX transition (49%), which required no code changes beyond model export. The llama.cpp transition involved both a model change

and an architectural change (in-process to HTTP sidecar), making it difficult to attribute the improvement to a single factor. The Azure transition reflects GPU-accelerated inference and higher-bandwidth batch processing. In production, the 44.3-second indexation time means that a newly published equity research report is searchable within one minute of upload.

3.3.3 Experiment 3: Retrieval Quality Ablation

To evaluate how chunking strategy and reranking affect retrieval quality, we constructed a domain-specific benchmark from the PrivilEdge fund prospectus (354 pages, French), a Luxembourg SICAV prospectus representative of the documents that fund screening operators query daily. The benchmark has 20 question-answer pairs modeled after real queries from fund screeners, covering fee structures, governance, investment restrictions, fund structure, taxation, and risk. Each question includes a ground-truth answer, the source page number(s), and a difficulty label. Table 3.4 shows representative examples. The full benchmark is in Appendix A and publicly available.[2]

Questions span four difficulty levels: *easy* (single-page factual lookup, 5 questions), *medium* (multi-paragraph reasoning, 6 questions), *hard* (cross-page comparison, 4 questions), and *very hard* (multi-hop retrieval across distant pages, 5 questions). The labels reflect retrieval complexity: an *easy* question may be hard to retrieve because the target fact is repeated across dozens of compartment-specific fee tables, diluting top-k precision.

Table 3.4: Representative benchmark questions by difficulty. GT Page indicates the ground-truth source page(s) in the 354-page PrivilEdge prospectus.

Difficulty	Question (French)	GT Page	Category
Easy	Quel est le montant minimum de souscription et de détention pour les Actions de classe U de PrivilEdge?	78	fee_structure
Medium	Quel est le seuil de rachat au-delà duquel la Société peut ajourner les remboursements, et quelle est la durée maximale?	59	operational_risk
Hard	Comparez la Commission de gestion maximale de la classe P entre PrivilEdge – Mondrian US Equity Value et PrivilEdge – Allianz All China Equity.	99, 121	fee_comparison
Very hard	Quelle est l'exposition attendue du compartiment PrivilEdge – Graham Quant Macro aux swaps sur rendement total (TRS) exprimée en somme des notionnels?	177	risk

We compare four retrieval configurations, all using Azure text-embedding-3-large and the same Elasticsearch index with $k = 300$:

- **Config A: Fixed-size chunking, no reranker.** TokenTextSplitter with 512-token chunks and 10% overlap. Page metadata preserved.

[2] https://huggingface.co/datasets/PeterAM4/priviledge-prospectus-rag-benchmark

- **Config B: Semantic + page-aware, no reranker.** SemanticSplitterNodeParser (buffer 8, threshold 99th percentile) with page metadata.
- **Config C_1: Semantic + page-aware + BGE-reranker-base.** Same as Config B with BGE-reranker-base (Xiao et al., 2023) (top 300 candidates).
- **Config C_2: Semantic + page-aware + Qwen3-Reranker-0.6B.** Same as Config B with Qwen3-Reranker-0.6B (Zhang et al., 2025) (8-bit quantized, multilingual, 100+ languages).

Evaluation metrics. *Context Hit Rate@K*: whether at least one of the top-K chunks originates from a ground-truth page ($K \in \{1,3,5,10,20\}$). *Mean Reciprocal Rank (MRR)*: the average of $1/r$ where r is the rank of the first relevant chunk. *Page Citation Accuracy*: whether the end-to-end pipeline (retrieval + GPT-5.1 generation) cites at least one ground-truth page. *Answer Accuracy*: whether the generated response contains the key facts from the ground-truth answer. All metrics are computed automatically. Answer Accuracy extracts two sets of atomic facts from the ground-truth answer: numeric facts (numbers, percentages, and French number words mapped to digits) and lexical facts (content words of five or more characters after filtering French stop words). Each extracted fact is checked as a substring of the lowercased generated response. A response passes if at least two-thirds of all extracted facts are found, with a relaxed fallback that also accepts responses matching at least half of the numeric facts and half of all facts combined. One question (Q14, Config C_2) required manual override where the scoring heuristic produced a false negative on a correct but concise response.

Results. Table 3.5 and Figure 3.4 present the full comparison.

Table 3.5: Retrieval quality ablation on 20 fund screening queries from the PrivilEdge prospectus. Context Hit Rate@K and MRR evaluate retrieval only. Page Citation and Answer Accuracy evaluate the end-to-end pipeline.

	Context Hit Rate@K						End-to-End	
Configuration	@1	@3	@5	@10	@20	MRR	Page Cite	Answer
A: Fixed-size, no reranker	40%	70%	70%	75%	85%	0.54	95%	95%
B: Semantic + page-aware	40%	60%	65%	75%	75%	0.53	95%	95%
C_1: + BGE-reranker	0%	5%	5%	10%	25%	0.05	85%	95%
C_2: + Qwen3-Reranker	**60%**	**70%**	**85%**	**95%**	**100%**	**0.69**	**95%**	**95%**

Config A vs. B: chunking strategy. Fixed-size chunking and semantic splitting achieve comparable retrieval quality, with Config A slightly ahead at Context Hit Rate@20 (85% vs. 75%). Both achieve identical end-to-end performance (95% page citation, 95% answer accuracy). For well-structured financial documents, the splitting strategy has less impact than the embedding model and the downstream context assembly.

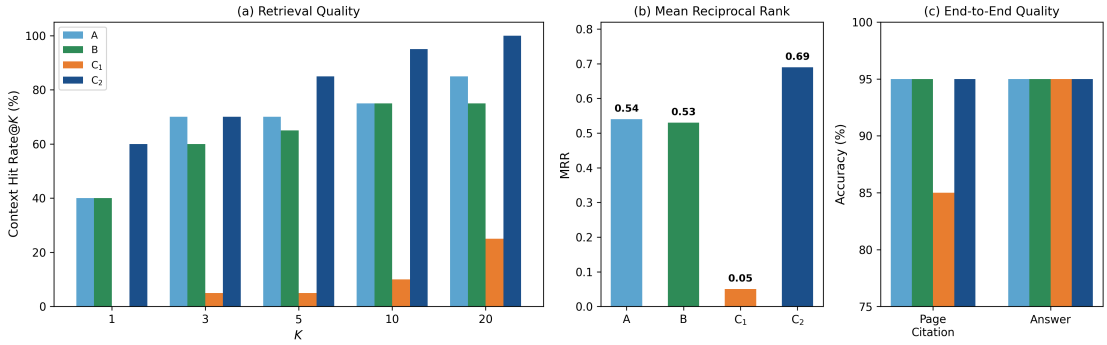

Figure 3.4: Retrieval quality ablation across four pipeline configurations. (a) Context Hit Rate@K shows that Config C_2 (Qwen3-Reranker) dominates at all K values, reaching 100% at K=20. Config C_1 (BGE-reranker) scores near zero due to its language mismatch on French text. (b) Mean Reciprocal Rank. (c) End-to-end Page Citation and Answer Accuracy remain high ($\geq$ 95%) for all configs except C_1's page citations (85%).

Config C_1: BGE-reranker on French text. Adding BGE-reranker-base severely degrades retrieval quality on this French-language benchmark: Context Hit Rate@5 drops to 5% and MRR to 0.05. Inspection of the reranked orderings shows that the model assigns near-zero relevance scores to most chunks, disrupting the sensible kNN ordering. This is a predictable consequence of the model being trained predominantly on English data, not a model defect—BGE-reranker-base performs well on English benchmarks. The production pipeline partially mitigates this because the reranker is only invoked when the full context exceeds the token limit (Algorithm 1).

Config C_2: Qwen3-Reranker as multilingual replacement. Replacing BGE with Qwen3-Reranker-0.6B (Zhang et al., 2025), a multilingual model supporting 100+ languages, resolves the language mismatch and produces the best results across all metrics. At K=20, Config C_2 achieves perfect retrieval: the correct page appears in the top 20 reranked chunks for every question. End-to-end metrics remain at 95%, confirming that the reranker improves retrieval precision without degrading generation quality.

Table 3.6 breaks down Config C_2 results by question difficulty.

Table 3.6: Config C_2 (Qwen3-Reranker) retrieval quality by question difficulty.

Difficulty	n	Context Hit Rate@5	MRR	Page Citation	Answer Acc.
Easy	5	3/5	0.51	5/5	5/5
Medium	6	6/6	1.00	5/6	6/6
Hard	4	3/4	0.40	4/4	4/4
Very hard	5	5/5	0.71	5/5	4/5
All	20	17/20 (85%)	0.69	19/20 (95%)	19/20 (95%)

Medium-difficulty questions achieve perfect HR@5 and MRR (1.00). Easy questions are paradoxically the hardest for retrieval (HR@5 = 3/5, MRR = 0.51) because the target content is repeated across dozens of compartment fee tables, diluting top-k precision. Very hard questions achieve HR@5 = 5/5 because their distinctive content produces fewer false-positive matches. The single Answer Accuracy miss (Q18, very hard) involves a cross-fund comparison where all four configurations produce incorrect fund names, suggesting a limitation of the retrieval approach itself.

3.4 Discussion

Quantization. The GGUF ablation shows that imatrix-calibrated K-quants achieve near-lossless compression for small embedding models. Q3_K_M-imat reduces model size by 70% with only +0.62 perplexity on financial text. Teams running embedding models on CPU-constrained infrastructure can quantize aggressively without meaningful quality loss, provided they calibrate the importance matrix on domain-representative text. The quality cliff below 4 BPW suggests that 4–5 BPW is the practical lower bound for production embedding models.

Indexation optimization. The 95.6% reduction in indexation time came from inference framework changes, not from algorithmic changes to the chunking pipeline. The same semantic splitting and page-aware chunking runs across all four configurations. The speedup comes entirely from how the embedding computation is executed.

Page-aware chunking. The page citation capability is a traceability feature that supports regulatory compliance. Under FINMA Guidance 08/2024, AI-generated information used in investment decisions must be traceable to its source. Page-level citations satisfy this requirement and, based on user feedback from fund screening operators, increase trust in the system's outputs.

Multilingual reranking. The BGE-reranker-base results on French text show that rerankers trained predominantly on English data can degrade retrieval quality on non-English documents. This is not a deficiency of the model itself but a language coverage limitation. Replacing it with Qwen3-Reranker-0.6B, a multilingual model, restored and exceeded baseline retrieval quality.

Limitations. First, pdfplumber (Singer-Vine, 2024) does not handle tables, charts, or images embedded in PDFs. Second, the document summarization step depends on the LLM's ability to compress financial text faithfully. Summarization errors can delay indexation.

4 LLM-Based Advisory Pipeline for Portfolio Recommendations

The RAG infrastructure presented in Chapter 3 provides the document grounding layer that enables bankers and fund screening operators to query the bank's internal research corpus. While that system addresses knowledge *retrieval*, the present chapter addresses knowledge *application*: given a client's portfolio and the bank's investment research, generate concrete buy and sell recommendations that a banker can present to the client.

4.1 Problem Formulation

At a private bank, investment intelligence flows at two levels. At the macro level, the Chief Investment Officer (CIO) publishes top-down views on asset allocation, regional exposure, and sector rotation, for instance, "overweight emerging-market equities" or "maintain a moderate pro-risk stance." At the micro level, equity research analysts issue bottom-up ratings on individual securities, ranging from Strong Buy to Sell, based on fundamental analysis including earnings forecasts, valuation models, and competitive positioning (Fieberg et al., 2025). Translating these two levels of research into concrete trades for a specific client portfolio is time-consuming. The banker must compare the CIO's views with the client's allocations, identify misalignments, select instruments from the bank's rated list, and size each trade appropriately. For a portfolio with 50 to 80 positions, this can take several hours.

We set out to build an advisory pipeline with four requirements: (1) recommendations must be grounded in the bank's own research rather than generic market commentary. (2) each recommendation must reference specific portfolio positions, with correct weights and realistic funding sources. (3) client portfolio data must remain entirely on-premise; and (4) the output must be structured and auditable, with separate banker-facing and client-facing text.

This chapter presents ALBA (Augmented Layer for Bespoke Advisory), a three-stage pipeline that distills the bank's research into structured context, enriches the client portfolio with allocation targets and deviation metrics, and generates structured advisory recommendations through a local LLM. We evaluate the system by comparing its output against two baselines:

GPT-5.1 with web search (no internal documents) and the production RAG system from Chapter 3 (automatic retrieval from the internal corpus running GPT-5.1 as the LLM). The evaluation uses three binary metrics that measure research alignment, CIO alignment, and portfolio grounding, following the same philosophy of reproducible, non-subjective evaluation applied in Chapter 3.

4.2 System Design

Figure 4.1 illustrates the end-to-end pipeline. The system operates in three stages: document distillation (cloud LLM), portfolio enrichment (deterministic computation), and advisory generation (on-premise LLM). The first two stages produce structured JSON files that the third stage consumes. This separation ensures that client portfolio data never leaves the on-premise environment, while document understanding benefits from a more capable cloud model.

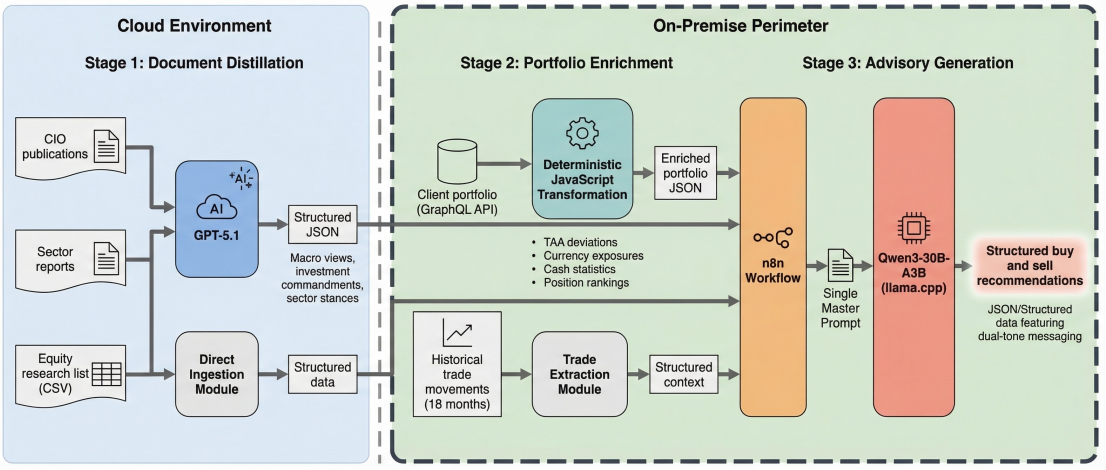

Figure 4.1: ALBA pipeline architecture. Stage 1 distills CIO publications and sector reports into structured JSON using GPT-5.1 (cloud, no client data). Stage 2 enriches the client portfolio with TAA targets and deviation metrics using deterministic JavaScript (no LLM). Stage 3 assembles all structured context into a single prompt and generates advisory recommendations via Qwen3-30B-A3B running on-premise through llama.cpp. The dashed boundary indicates the on-premise perimeter: only Stage 3 and the portfolio enrichment touch client data.

4.2.1 Stage 1: Document Distillation

The bank's CIO publishes research continuously: weekly investment notes, CIO office viewpoints, flash reports on geopolitical events, and investment committee updates. In a typical month, 10 to 15 such documents are published. Rather than retrieving chunks at query time (as in Chapter 3), ALBA distills each document into a structured JSON representation before the advisory prompt is assembled.

For CIO publications, each PDF is retrieved from Elasticsearch by content type and language.

The document text is sent to GPT-5.1 with a structured extraction prompt that produces a JSON object containing: document metadata (title, date, type, risk stance), a macro view summary, a structured view encoding the CIO's stance on equity regions, countries, sectors, and currencies, and a list of investment commandments. Each commandment carries a title, the underlying asset or theme, a stance (overweight, underweight, neutral, prefer), tags for programmatic matching, a rationale, and references to the source documents. The schema enforces null values for unknown fields rather than empty strings, preventing the downstream model from treating absence of a view as a neutral stance.

A second aggregation step merges the individual document JSONs into a single aggregated view. This step deduplicates commandments that appear across multiple publications, resolves temporal conflicts where later documents supersede earlier ones, and consolidates the structured view into a single object. The aggregated CIO view for the evaluation period contains 32 commandments drawn from 10 source documents spanning three weeks of publications.

Sector comments and the equity research list follow simpler paths. Sector comments are extracted from the equity research department's monthly sector report, producing per-sector summaries with analyst names, overall stance, and key observations. The equity research list is a structured file containing 255 rated stocks with ISINs, ratings (Strong Buy, Buy, Hold, Sell), price targets, valuation multiples, and thematic tags. This list is ingested directly as structured data without LLM distillation.

4.2.2 Stage 2: Portfolio Enrichment

The client portfolio arrives from the bank's portfolio management system via a GraphQL API (GraphQL Foundation, 2025) as a flat list of positions, each carrying a valuation, asset label, ISIN, quotation currency, risk currency, tactical asset allocation (TAA) bucket, and weight. A second API call retrieves the portfolio's benchmark TAA target weights and currency allocation targets.

A deterministic JavaScript transformation enriches this raw data into a structured JSON object containing: the portfolio identifier (anonymized), base currency, total value, average position weight, cash statistics, exposure deviations by TAA bucket (target, actual, and deviation for each bucket), exposure deviations by risk currency, the largest overweights and underweights sorted by magnitude, and the full position list. No LLM is involved: the deviations are computed arithmetically from the target and actual weights.

This precomputation is a deliberate design choice. The advisory prompt instructs the LLM to trust the precomputed metrics and not to recalculate portfolio statistics. This prevents a failure mode observed in early prototypes where the LLM would hallucinate portfolio weights or miscompute deviations from the position list, producing recommendations based on incorrect allocation data.

Historical trade movements over the preceding 18 months are extracted separately, producing a structured object that captures the direction, size, and timing of recent trades per instrument. This context enables the LLM to identify behavioral patterns and produce recommendations consistent with the portfolio's trajectory.

Evaluation portfolios. We evaluate the system on three test portfolios constructed to span different base currencies, sizes, and structural characteristics. Portfolio 1 is a USD-denominated portfolio with 81 positions across 14 TAA buckets and an average position weight of 1.23%, exhibiting heavy concentration in Eurozone financials, zero emerging-market equity allocation despite a 7.1% TAA target, and US equity exposure approximately half the target. Portfolio 2 is a CHF-denominated portfolio with 41 positions, dominated by global equity funds and physical gold, with large underweights in Swiss equities and corporate bonds. Portfolio 3 is a CHF-denominated portfolio with 138 positions including substantial private equity commitments, a massive gold overweight (+17%), and US equity exposure 14 percentage points below target. Table 4.1 shows the 20 largest equity positions in Portfolio 1, which we use as the detailed example throughout this section.

Table 4.1: Top 20 equity positions in Portfolio 1 (USD, 81 positions), sorted by weight. The portfolio contains 45 equity positions totaling 75.4%, with the remaining 24.6% in bonds, cash, and fiduciary deposits. TAA bucket labels: Eq.EMU = Eurozone equities, Eq.US = US equities, Eq.CH = Swiss equities, Eq.JP = Japanese equities, Eq.SC = small/mid-cap equities, Eq.Oth = other equities.

Weight	Position	ISIN	Bucket	Ccy
4.72%	Deutsche Bank	DE0005140008	Eq.EMU	EUR
4.10%	Kinross Gold	CA4969024047	Eq.Oth	CAD
4.08%	Orange	FR0000133308	Eq.EMU	EUR
4.03%	Teva Pharmaceutical	US8816242098	Eq.Oth	ILS
3.74%	Commerzbank	DE000CBK1001	Eq.EMU	EUR
3.67%	ENI	IT0003132476	Eq.EMU	EUR
3.47%	AT&T	US00206R1023	Eq.US	USD
3.25%	Panasonic Holdings	JP3866800000	Eq.JP	JPY
2.73%	Roche	CH0012032113	Eq.CH	CHF
2.43%	BNP Paribas	FR0000131104	Eq.EMU	EUR
2.30%	Gilead Sciences	US3755581036	Eq.US	USD
2.16%	Novartis	CH0012005267	Eq.CH	CHF
2.12%	DXC Technology	US23355L1061	Eq.SC	USD
2.03%	Viatris	US92556V1061	Eq.SC	USD
2.01%	Whirlpool	US9633201069	Eq.SC	USD
1.85%	Société Générale	FR0000130809	Eq.EMU	EUR
1.74%	Swatch Group	CH0012255151	Eq.CH	CHF
1.73%	Jazz Pharmaceuticals	IE00B4Q5ZN47	Eq.SC	USD
1.61%	Bunge Global	CH1300646267	Eq.US	USD
1.58%	Stadler Rail	CH0002178181	Eq.SC	CHF

4.2.3 Stage 3: Context Assembly and Generation

The final stage assembles all structured context into a single prompt and sends it to a local LLM for generation. An n8n (n8n GmbH, 2025) workflow orchestrates this process: it lists all structured files from three directories (CIO views, asset research, and portfolio data), reads each file, formats its contents, and appends them sequentially to a master prompt. The assembled prompt spans approximately 120,000 tokens and is sent via HTTP POST to a llama.cpp server running Qwen3-30B-A3B (Yang et al., 2025), a mixture-of-experts model (Shazeer et al., 2017) with 30 billion total parameters and 3 billion active parameters per forward pass. At the inference speed of Qwen3-30B-A3B on the bank's CPU hardware, generating the full advisory output for a single portfolio requires approximately four hours (using 20 GB of RAM and 8 CPU cores).

The model selection reflects a deliberate trade-off. Qwen3-30B-A3B achieves competitive reasoning quality with a small active parameter count, enabling on-premise inference on hardware available within the bank's data centers. The model runs in thinking mode, which allows it to reason through the macro-to-micro decision process before producing the structured output.

The master prompt (reproduced in Appendix D) encodes a three-level reasoning hierarchy that mirrors professional portfolio management practice:

1. **Macro level (CIO/IC).** The model identifies the CIO's current preferences on asset classes, regions, styles, and alternatives, and defines one or two macro themes to implement.
2. **Portfolio structure.** Using the precomputed exposure deviations, the model identifies where the portfolio is underweight in areas the CIO favors and overweight in areas the CIO is cautious on.
3. **Instrument selection.** Only after the macro and structural picture is established does the model select specific instruments from the equity research list to implement the chosen themes.

The prompt enforces several constraints to prevent common failure modes. Attribution discipline requires the model to distinguish between CIO views, TAA targets, and portfolio observations, and to never claim the CIO "emphasized" a view unless the text explicitly states it. Cash-aware sell logic permits sell recommendations only when cash is insufficient to fund the proposed buy. Sizing discipline anchors the recommended buy weight to the portfolio's average position weight.

The output is a structured JSON object organized into strategic axes (CIO/TAA alignment, strategic themes, high-conviction micro opportunities, relative-value switches, and portfolio risk), each containing one or more investment ideas with buy recommendations (ISIN, weight, rating, key metrics, house view alignment) and funding sources (specific positions to trim with

pre-trade and post-trade weights). Dual-tone messaging produces separate banker-facing text (technical) and client-facing text (plain language).

4.3 Experimental Evaluation

4.3.1 Evaluation Design

Evaluating generated financial recommendations poses a fundamental challenge: there is no ground truth. Unlike the retrieval task in Chapter 3, where each question has a known correct answer page, an investment recommendation can be "good" along multiple dimensions without a single correct answer. We avoid LLM-as-judge approaches to maintain consistency with the binary, reproducible evaluation methodology used throughout this thesis.

Instead, we evaluate whether recommendations are *actionable within the bank's investment framework*: the same standard a human banker must meet. Under FINMA suitability obligations, a banker cannot recommend a stock without internal research coverage, cannot justify a trade that contradicts the CIO's published house view, and cannot propose selling a position the client does not hold. These are verifiable facts, not subjective judgments. A recommendation to buy a stock that the bank's own analysts have not rated is unusable regardless of whether the stock subsequently rises or falls.

4.3.2 Conditions

We compare three configurations representing increasing levels of context specificity. All three receive the same portfolio data and are prompted to generate structured buy/sell recommendations. Each condition is evaluated on all three test portfolios described in Section 4.2.2.

- **Condition A: GPT-5.1 with web search.** The portfolio is provided as structured text. The model has access to public information via web search but receives no internal documents.
- **Condition B: GPT-5.1 with RAG retrieval.** The portfolio is provided alongside the production RAG system from Chapter 3. The model retrieves context automatically from the bank's indexed document corpus based on embedding similarity.
- **Condition C: ALBA (expert-curated context).** The full ALBA pipeline. The portfolio is accompanied by the aggregated CIO view (32 commandments from 10 source documents), structured sector comments, the 255-stock equity research list, and 18 months of trade history. Generation uses Qwen3-30B-A3B (3B active parameters) rather than GPT-5.1.

A notable asymmetry: Condition C uses a significantly smaller model but receives richer, domain-specific context.

4.3.3 Metrics

Each condition produces 7 to 10 buy/sell ideas per portfolio. We score each idea on three binary metrics (0 or 1), then report the pooled pass rate across all ideas and portfolios. The scoring is automated by a Python script that cross-references each recommendation against the bank's equity research list, the aggregated CIO view, and the actual portfolio holdings.

CIO commandments. Table 4.2 lists representative commandments from the aggregated CIO house view. Each commandment encodes a stance on a specific region, sector, or theme, expressed as structured tags for programmatic matching. The full set contains 32 commandments drawn from 10 source documents.

Table 4.2: Representative CIO commandments used for alignment scoring. Each commandment carries a stance and structured tags. The full set contains 32 commandments.

Commandment	Stance	Tags
Overweight Swiss equities	overweight	`country:switzerland`
Overweight global healthcare	overweight	`sector:healthcare`
Overweight EM equities vs. DM	overweight	`region:em`
Prefer quality growth tech / AI	prefer	`sector:information_technology, theme:ai`
Underweight consumer staples	underweight	`sector:consumer_staples`
Underweight US equities	underweight	`country:us`

Research Alignment. Whether the buy recommendation's ISIN appears in the bank's 255-stock equity research list with a Buy or Strong Buy rating.

CIO Alignment. Whether the buy recommendation is consistent with the CIO's house view. The scoring script maps each recommendation's region, sector, and TAA bucket to the tag vocabulary shown in Table 4.2 and checks for alignment in two steps. First, it searches for at least one commandment with a supportive stance (overweight or prefer) that shares a tag with the recommendation. Second, it checks whether a more specific commandment with a blocking stance (underweight or avoid) contradicts the match. For example, a recommendation to buy Novartis (Swiss, healthcare) passes because it matches both the overweight on Swiss equities and the overweight on healthcare. A recommendation to buy Nestlé (Swiss, consumer staples) fails because the underweight on consumer staples blocks the Swiss equity support. A recommendation to buy Swiss Prime Site (real estate) fails because no commandment addresses real estate.

Portfolio Grounding. Whether the sell/trim recommendations reference positions that actually exist in the portfolio with approximately correct weights (within 1.5 percentage points).

Overall Quality.requires all three criteria to pass simultaneously.

4.4 Results

Table 4.3 and Figure 4.2 present the pooled comparison across all three conditions and three portfolios. Condition A produces 10 ideas per portfolio (30 total), Condition B produces 10 per portfolio (30 total), and Condition C produces 7 to 10 per portfolio (25 total) depending on the number of actionable axes identified by the model.

Table 4.3: Advisory quality pooled across three test portfolios. Research Alignment measures whether buy recommendations appear in the bank's rated equity list with Buy or Strong Buy. CIO Alignment checks consistency with the 32 published commandments. Portfolio Grounding verifies that sells reference actual positions with correct weights. Overall Quality requires all three.

Metric	A: GPT-5.1 + Web	B: RAG	**C: ALBA**
Research Alignment	14/30 (47%)	28/30 (93%)	**24/25 (96%)**
CIO Alignment	25/30 (83%)	30/30 (100%)	**25/25 (100%)**
Portfolio Grounding	30/30 (100%)	30/30 (100%)	25/25 (100%)
Overall Quality	14/30 (47%)	28/30 (93%)	**24/25 (96%)**

Condition A: GPT-5.1 with web search. Across the three portfolios, 16 of 30 buy recommendations fail Research Alignment. The failure modes are consistent: the model recommends stocks the bank does not cover (Berkshire Hathaway, ICICI Bank, PSP Swiss Property, Keyence), stocks rated Hold rather than Buy (Novartis, ABB, Caterpillar, Swiss Re), and instruments under the wrong ISIN (TSMC via the US-listed ADR rather than the Taiwan-listed shares the bank rates). CIO Alignment reaches 83% because web search surfaces general market consensus that partially overlaps with the bank's views. On Portfolio 1, for instance, the model recommends TSMC to close the emerging-market gap, which aligns with the CIO's EM overweight, but uses the wrong share class. This illustrates how generic financial knowledge differs from institutional knowledge: recommending the wrong share class of the right company creates settlement and regulatory complications.

Condition B: RAG with automatic retrieval. Access to the indexed document corpus lifts Research Alignment from 47% to 93%. The RAG system retrieves actual CIO publications and equity flash reports, guiding the model toward the bank's investment universe. CIO Alignment reaches 100% across all three portfolios. The two Research Alignment failures (both on Portfolio 1) are Eaton and Prysmian, rated Hold rather than Buy. The RAG system retrieves documents that discuss these companies favorably but cannot distinguish between positive discussion and a formal Buy rating, since it does not have access to the structured research list with explicit ratings.

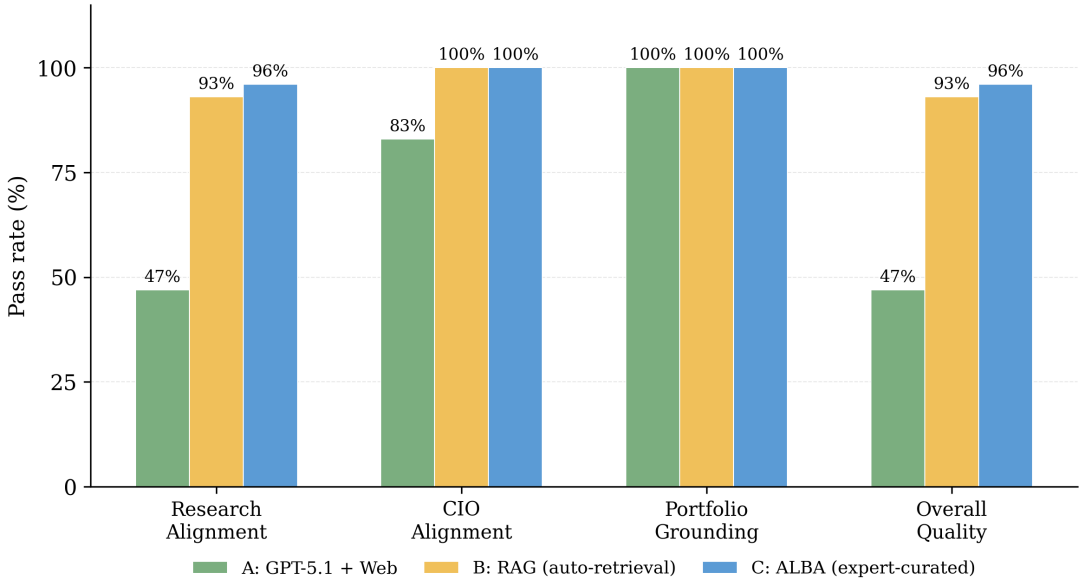

Figure 4.2: Advisory quality pooled across three test portfolios. Research Alignment improves from 47% (web search) to 93% (RAG) to 96% (ALBA). CIO Alignment rises from 83% to 100% once the model has access to internal CIO publications. Portfolio Grounding is 100% for all conditions. The Overall Quality progression (47% → 93% → 96%) confirms that curated context consistently outperforms both web search and automatic retrieval across diverse portfolio profiles.

Condition C: ALBA.ALBA achieves the highest scores across all three portfolios despite using a significantly smaller model (3B active parameters) than GPT-5.1. Research Alignment reaches 96%: 24 of 25 buy recommendations correspond to stocks rated Buy or Strong Buy. The single miss is Shell PLC on Portfolio 1, recommended as a relative-value switch within the energy bucket rather than as a research-driven idea. Shell does not appear in the bank's rated list, but in practice a banker would accept the trade since the switch logic is sound. CIO Alignment and Portfolio Grounding are both 100% across all portfolios. Table 4.4 shows the per-idea breakdown for Portfolio 1. A full comparison of the TSMC recommendation from Condition A and Condition C is in Appendix C.

Table 4.4: ALBA (Condition C) per-idea scoring on Portfolio 1. R = Research Alignment, C = CIO Alignment, G = Portfolio Grounding. Portfolios 2 and 3 achieve 7/7 and 8/8 on all three metrics respectively.

ID	Buy recommendation	Axis	Rating	R	C	G
1	Taiwan Semiconductor (TSMC)	CIO/TAA Alignment	Strong Buy	✓	✓	✓
2	PDD Holdings	CIO/TAA Alignment	Strong Buy	✓	✓	✓
3	Bank of America	CIO/TAA Alignment	Strong Buy	✓	✓	✓
4	Microsoft	Strategic Themes	Strong Buy	✓	✓	✓
5	Veolia Environnement	Strategic Themes	Buy	✓	✓	✓
6	Thermo Fisher Scientific	Micro Opportunities	Strong Buy	✓	✓	✓
7	Intercontinental Exchange	Micro Opportunities	Strong Buy	✓	✓	✓
8	Shell PLC	Relative Value	—	✗	✓	✓
9	Royal Gold	Relative Value	Buy	✓	✓	✓
10	ServiceNow	Relative Value	Strong Buy	✓	✓	✓
Total				9/10	10/10	10/10

4.5 Discussion

Context quality vs. model capability. The central finding is that ALBA, using a 3B-active-parameter model, outperforms GPT-5.1 on domain-specific advisory quality when provided with richer, curated context. Across three test portfolios with different base currencies, sizes, and structural characteristics, the Overall Quality progression from 47% (web search) to 93% (RAG) to 96% (ALBA) is consistent. For structured financial reasoning, the bottleneck is not the model's general capability but the quality and specificity of the context it receives. The bank's 255-stock research list, 32 CIO commandments, enriched portfolio deviations, and 18-month trade history provide the model with precisely the information a human portfolio manager would use. A more capable model without this context produces recommendations that are well-reasoned but not actionable within the bank's investment framework.

Curated structured data vs. automatic retrieval. The comparison between Conditions B and C isolates the effect of context curation. Both achieve 100% CIO Alignment across all portfolios, confirming that access to internal CIO documents is the critical factor for macro

alignment. The difference appears in Research Alignment (93% vs. 96%). RAG's failures on Portfolio 1 (Eaton and Prysmian, both rated Hold) illustrate a specific limitation of retrieval-based context. The system surfaces documents that discuss these companies favorably but cannot distinguish between positive discussion and a formal Buy rating. ALBA avoids this error because the structured research list includes explicit ratings alongside each ISIN. For tasks requiring precise categorical information (ratings, price targets, valuation multiples), curated structured data outperforms chunk-based retrieval.

Privacy-preserving hybrid architecture.ALBA's three-stage design separates document understanding (cloud) from portfolio-specific generation (on-premise). Stage 1 processes only the bank's published research, which contains no client data, using GPT-5.1 via the Azure API. Stage 3 processes the client portfolio using Qwen3-30B-A3B running locally through llama.cpp. The strongest available model handles the task that does not touch client data, while a smaller but locally deployable model handles the privacy-sensitive generation step.

Structured prompting as reasoning scaffold. The master prompt's three-level hierarchy (Macro → Structure → Instruments) is not merely a formatting convention but a reasoning scaffold. Early prototypes that provided the same context without the hierarchical instruction produced recommendations that jumped directly to instrument selection, often recommending stocks that were individually attractive but misaligned with the CIO's macro stance or that worsened existing portfolio overweights. The explicit instruction to establish the macro theme before selecting instruments mirrors the discipline that experienced portfolio managers follow.

Limitations. The evaluation spans three test portfolios with diverse structural profiles. While the results are consistent across all three, evaluating on a larger sample would further strengthen generalization claims. The distilled context has a shelf life of approximately two to four weeks, after which new CIO publications must be processed. The most significant operational constraint is inference latency, generating the full output requires approximately four hours on CPU hardware. The bank has since approved sending anonymized portfolios to the cloud, which will reduce this to minutes while preserving the same three-stage architecture.

5 Regression-Based Anomaly Detection in Tax Statements

5.1 Problem Formulation

As introduced in Chapter 1, the bank produces annual tax statements for Swiss-resident clients containing over 100,000 lines of structured data per statement. These statements must be cross-referenced against ICTax (Income & Capital Taxes), the official database published by the Swiss Federal Tax Administration (AFC) at `ictax.admin.ch`.[1] ICTax contains security valuations, fiscal exchange rates, and dividend payment records for every security traded in Switzerland, serving as the authoritative reference for tax reporting. Certain fields within the statements are imputed manually around dividend events, introducing data-entry errors. We have no labeled anomaly dataset and are not detecting intentional fraud but accidental errors. We frame the problem as unsupervised anomaly detection via regression: learn expected column relationships from historical data and flag cases where production residuals deviate from the training distribution.

5.2 System Design

Figure 5.1 illustrates the end-to-end pipeline. The system consists of a preprocessing pipeline, a training pipeline that discovers column relationships via ElasticNet, and an inference service that scores new statements against the learned models. It is deployed as a Flask API behind TLS with Prometheus monitoring, exposing endpoints for batch training and per-statement detection.

5.2.1 Data Format and Parsing

Tax statements arrive as XML documents from the bank's document generation system (DocGen). Each XML encodes a hierarchical structure of tables (`TableauLibre`), nested five levels deep (`Niveau1` through `Niveau5`), with rows containing indicator–value pairs. A prior step

[1] https://www.ictax.admin.ch

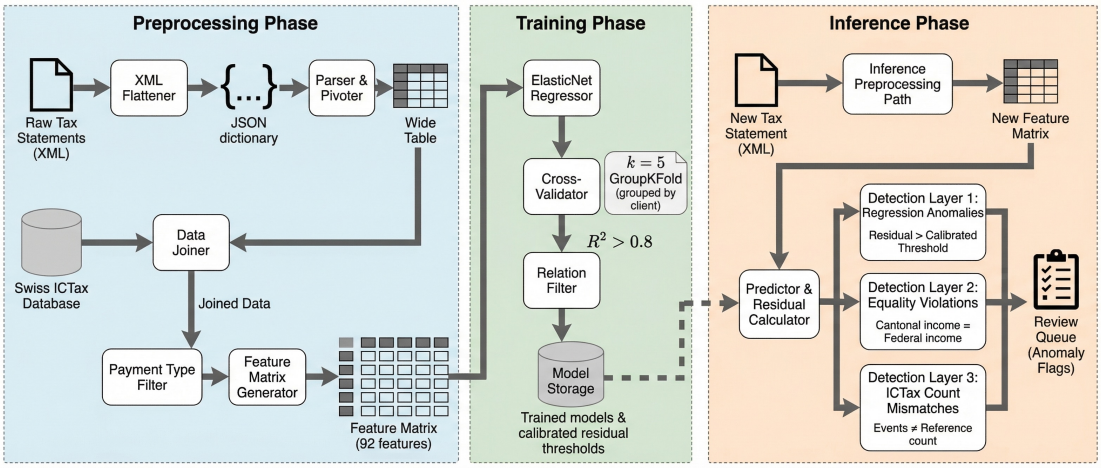

Figure 5.1: Anomaly detection pipeline. The preprocessing phase flattens XML tax statements into a JSON dictionary, parses and pivots the hierarchical keys into a wide table, joins with the Swiss ICTax reference database, filters non-cash payment types, and constructs the feature matrix. The training phase fits an ElasticNet regressor per target column using 5-fold grouped cross-validation by client, retaining only relations with $R^2 > 0.8$. The inference phase applies the same preprocessing to a new statement and routes predictions through three detection layers: regression residual thresholds, equality violations, and ICTax payment count mismatches.

flattens each XML into a JSON dictionary of approximately 5,000 key–value pairs. Keys encode the full hierarchy as a dot-separated path:

```
TableauLibre.17.niv1.0.niv2.0.niv3.0.niv4.0.niv5.6.row.7
    .CH_GLOB_MIN_04_AmountICR.number -> 250.0
```

Here, table 17 at nesting level niv5 subgroup 6 groups all data for a single ISIN. Row 7 is a specific dividend payment. The field `AmountICR` corresponds to *Rendement impôt cantonal CHF* (cantonal tax income in CHF) with value 250.0. The preprocessor parses each key, pivots row-level data into a wide table where each row represents a payment event, and merges total-level entries (which carry the ISIN) to associate each payment with its security.

5.2.2 Feature Matrix Construction

The pipeline retains columns matching a curated set of root indicators relevant to dividend taxation. Table 5.1 maps the code indicators to their official labels from the XML `Libellés` section.

After pivoting, the pipeline aggregates multiple payment rows per ISIN per day and joins with the ICTax reference database on the (ISIN, payment date) composite key. The ICTax join adds the reference payment value in CHF, exchange rate, currency, and sign field. From this, binary indicators are derived: `currency_conflict` (document vs. ICTax currency mismatch),

Table 5.1: Tax statement columns used as features and targets. Labels are extracted from the `Libellés` section of the source XML. The six amount columns (top group) serve as regression targets; the remaining columns provide context features.

Code indicator	Official label (FR)	English description
`AmountIFR`	Rendement impôt fédéral direct	Federal tax income
`AmountICR`	Rendement impôt cantonal	Cantonal tax income
`AmountFTC`	Impôt étranger non récupérable (DA-1)	Non-recoverable foreign tax
`AmountIA`	Rendement soumis à l'impôt anticipé	Withholding tax base
`AmountIC-2.5`	—	Supplementary tax at 2.5%
`AmountOWB1`	Retenue supplémentaire d'impôt (Rsi)	Additional withholding (Rsi)
`XRate`	Cours fiscal / Cours de change	Tax exchange rate
`COL02a`	Valeur nominale ou nombre de titres	Nominal value or share count
`COL06`	Fortune imposable CHF	Taxable assets in CHF
`Description`	Désignation	Position description (date, amount, currency)

`sign_has_kep` (capital repayment), `sign_has_kg` (bonus shares), and missingness flags for exchange rate and currency.

Payment type filtering. Three categories of ICTax payments are excluded before feature construction because they follow different statistical patterns than regular dividends: stock splits (`gratis` = 1, share quantity changes with no cash flow), stock dividends (`variant` = 2, additional shares rather than cash), and deleted entries (corrections). Including these was the primary source of false positives in early iterations, as discussed in Section 5.4.4.

The final feature matrix contains one row per (client, ISIN, payment date) triplet. The production training dataset yields 160,461 rows across 4,267 client groups.

5.3 Methodology

5.3.1 Relation Discovery via ElasticNet

The system treats each numeric column as a potential target variable and attempts to predict it from the remaining columns. A target is considered for modeling only if it has at least 30 non-null rows across at least 3 distinct client groups, and is not constant. If its cross-validated R^2 exceeds 0.8, the relationship is retained as a model, deviations at inference time are flagged as anomalies.

Of the six target columns in Table 5.1, the non-recoverable foreign tax (AmountFTC) did not exceed the R^2 threshold and was excluded from the production model set.

Feature pipeline. The feature matrix is processed through a `ColumnTransformer` with three parallel branches. Numeric features (amount columns, exchange rate, ICTax reference value)

pass through an imputer, a degree-2 polynomial expansion with interaction terms only (`interaction_only=True`), and a standard scaler without centering to preserve sparsity. The polynomial expansion is essential: multiplicative relationships are natural in tax data—for example, the withholding amount in CHF equals the gross dividend in foreign currency times the exchange rate. Binary indicator features (missingness flags, currency conflict, sign type) pass through a most-frequent imputer. Categorical features (currency, ICTax sign) pass through a constant imputer (filling missing values with `"MISSING"`) followed by one-hot encoding with the first category dropped to avoid collinearity.

Nested cross-validation. We use ElasticNet (Zou and Hastie, 2005) (Equation 2.2 in Chapter 2) with a nested cross-validation scheme. The outer loop uses `GroupKFold` with $k = 5$ folds grouped by client identifier (`doc_id`), ensuring that all rows from the same client appear in the same fold. This prevents the model from exploiting client-specific correlations that do not generalize. Within each outer fold, `ElasticNetCV` selects the regularization parameters λ and α from eight L1 ratios ($\alpha \in \{0.05, 0.1, 0.3, 0.5, 0.7, 0.9, 0.95, 0.99\}$).

After evaluating on held-out folds, the final model is refit on the full dataset. The out-of-fold residuals from the outer loop are preserved for threshold calibration (Section 5.3.3), avoiding the optimistic bias that would result from calibrating on in-sample residuals.

Collinearity detection. The pipeline discovers column pairs with Pearson correlation above 0.999 and drops the lower-variance column from each pair before training. It also identifies column pairs that are exactly equal (within floating-point tolerance) across all observations. These equality constraints are stored alongside the regression models and enforced as a separate detection layer at inference time.

5.3.2 Imputation

Missing numeric values represent genuinely absent amounts (e.g., no foreign tax credit for a given security). The production configuration imputes with zero, reflecting domain semantics. As an alternative, we implemented a rank-8 matrix factorization imputer inspired by softImpute (Mazumder et al., 2010), iterating truncated SVD until convergence ($\tau = 10^{-4}$, max 15 iterations). Both strategies are compared in Section 5.4.

5.3.3 Anomaly Scoring

At inference, each model predicts the target column and computes the relative residual:

$$r_i = \frac{|y_i - \hat{y}_i|}{\max(|y_i|, \epsilon)}, \quad \epsilon = 10^{-6} \tag{5.1}$$

Relative residuals handle the fact that tax amounts span orders of magnitude. During training, the system calibrates detection thresholds from the out-of-fold residual distribution at five quantiles (0.90, 0.95, 0.98, 0.99, 0.995). At inference, the system iterates from the strictest threshold to the most lenient. The first threshold exceeded determines the anomaly's confidence level. If the residual falls below all thresholds, the observation is not flagged.

Before running the regression models, the system checks whether each model's required input columns are present and non-null in the statement. Models that cannot run due to missing columns are reported as incomplete relations in the API response, allowing the calling system to distinguish "no anomaly detected" from "detection was not possible."

Two additional detection layers operate independently. An **ICTax payment count check** compares the number of payment events in the client's statement against the ICTax reference on shared ISINs; mismatches signal missing or duplicated entries. This check was later disabled in production due to a high false positive rate, as discussed in Section 5.4.3. An **equality check** enforces column pairs discovered as identical during training—in our data, cantonal income (AmountICR) and federal income (AmountIFR) are equal across all 160,461 rows, reflecting a regulatory invariant for Swiss fiscal residents.

5.4 Results

5.4.1 Feature Correlation Structure

Figure 5.2 shows the pairwise scatter plots and marginal distributions for the six target columns plus the exchange rate and ICTax reference value. Three patterns are visible: cantonal and federal income are perfectly aligned ($r = 1.0$), confirming the regulatory equality the system discovers automatically; several column pairs show moderate correlations that the regression models learn to exploit. Most distributions are heavily right-skewed, with the bulk of observations near zero. The exchange rate shows distinct cluster structures corresponding to major currency pairs (CHF, EUR, USD), and the interaction terms in the polynomial expansion are essential for capturing the multiplicative dependencies between amounts and rates.

5.4.2 Discovered Relations

Training discovers five column relationships passing the $R^2 \geq 0.8$ threshold, plus the equality constraint. Table 5.2 summarizes the metrics under both imputation strategies.

All models achieve $R^2 > 0.93$. The ElasticNet consistently selects $\alpha = 0.99$, producing sparse models: the additional withholding model uses only 10 nonzero coefficients out of 92 features. Top coefficients are interaction terms between amount columns and exchange rates, reflecting the multiplicative dependencies visible in Figure 5.2. The identical metrics for cantonal and federal income are expected given the perfect correlation; the collinearity check drops one when training models for other targets.

Figure 5.2: Pairwise scatter matrix for 8 of 10 numeric features from Table 5.1 (AmountIA and AmountOWB1 omitted for readability). Diagonal panels show marginal distributions. The perfect alignment between AmountICR and AmountIFR confirms the regulatory equality discovered by the system. Most distributions are heavily right-skewed.

Table 5.2: Discovered relations and cross-validated performance (5-fold `GroupKFold` by client). Config A: constant (zero) imputation. Config B: rank-8 matrix factorization. *NZ*: nonzero coefficients after regularization.

	Config A (constant)			**Config B (MF)**		
Target	CV R^2	CV MAE	NZ	CV R^2	CV MAE	NZ
Suppl. tax 2.5%	0.953 ± 0.050	131.8	26	0.953 ± 0.050	131.8	26
Withholding base	0.935 ± 0.064	1,420.6	38	0.935 ± 0.064	1,420.6	38
Cantonal income	0.974 ± 0.024	1,259.6	49	0.974 ± 0.024	1,259.6	49
Federal income	0.974 ± 0.024	1,259.6	49	0.974 ± 0.024	1,259.6	49
Additional wh. (Rsi)	0.932 ± 0.072	16.7	10	0.932 ± 0.072	16.7	10
Equality: cantonal income ≡ federal income (160,461/160,461 rows, tol = 0.0)						

Config B (matrix factorization) produces identical results to Config A across all five targets: R^2 values match to six decimal places and the same coefficients are selected. This is expected, when a security has no foreign tax credit, the amount is genuinely zero, not an unobserved latent value. The MF imputer converges to near-zero values for these entries, reproducing what constant imputation does directly. We use constant imputation in production.

5.4.3 Production Deployment

The system processes 21,679 client statements. Approximately 2% are flagged as containing anomalies. After review by the tax operations team, 10 anomalies are escalated, of which 8 are confirmed genuine errors (80% precision on escalated cases). We present representative examples below. Client identifiers are omitted.

Example 1: Exchange rate unit error (JPY).A JPY-denominated Luxembourg fund (ISIN: LU2191964076) reports an exchange rate of 0.51296 in the tax statement. The ICTax reference lists 0.005129 for the same position. The ratio is almost exactly 100: the rate was entered in centimes (Rappen) rather than francs. The regression model on cantonal income flags this because the predicted income, derived from the nominal amount and exchange rate interaction term, is two orders of magnitude lower than the reported value.

Example 2: Exchange rate unit error (CNY).A BYD position (ISIN: CNE100000296) reports an exchange rate of 11.25 CHF/CNY. The actual CNY/CHF rate is approximately 0.114, again a factor-of-100 discrepancy. The same error pattern recurs on a subsequent dividend payment for the same position, confirming that exchange rate centimes errors are a systematic failure mode in manual entry.

Example 3: Micro-amount foreign tax flag.A Luxembourg-domiciled fund shows a non-recoverable foreign tax (DA-1) of CHF 0.02. The regression model flags this because the predicted DA-1 amount is zero: the fund's ICTax reference indicates no foreign tax should apply. The CHF 0.02 is a rounding artifact from a currency conversion that should have been zeroed out.

Example 4: ICTax payment count mismatch. The payment count check identified approximately four genuine mismatches, including positions in NVIDIA and Alphabet where the number of dividend events did not match the ICTax reference. However, the check produced a high false positive rate due to two structural issues and was disabled in production. First, a client's positions for a single ISIN can span multiple tables in the statement, and the aggregation step does not always align with ICTax's single entry per payment date. Second, the model does not account for temporal coverage: partial-year clients (mid-year domicile changes, position entries/exits) have fewer payment events than ICTax lists for a full year, triggering false mismatches.

5.4.4 False Positive Rate Progression

The initial deployment had a false positive rate of approximately 90%. The reduction to approximately 5% came from preprocessing, not model changes:

1. **Payment type filtering.** Stock splits, stock dividends, and capital repayments follow different patterns than cash dividends. Including them violated the assumption that column relationships are consistent across rows.
2. **Daily aggregation.** Multiple payment components on the same date for the same ISIN produced partial-amount rows that inflated residuals.
3. **ICTax sign and currency features.** Adding the payment sign and currency conflict indicator as features let the model learn context-dependent relationships.

This illustrates that in production anomaly detection, the most impactful improvements come from domain understanding rather than model architecture.

5.5 Discussion

The system requires no labeled anomalies and produces interpretable outputs: for each flag, the model coefficients indicate which columns drove the prediction. Inference takes approximately 7 seconds per statement, dominated by the preprocessing and ICTax join rather than the regression predictions.

The main limitation is the absence of temporal information. The model has no representation of holding periods, fiscal domicile changes, or position entry/exit dates. This produces false positives for partial-year positions and was the reason the payment count check was disabled despite catching genuine errors. Adding temporal coverage features would address both the regression and count-based false positives simultaneously. The system also assumes stationarity: regulatory changes or new security types could invalidate learned models. Systematic feedback collection over a full fiscal year would provide a larger sample for precision estimation.

6 Discussion and Conclusion

This chapter presents the deployed system as a whole, compares the three AI modalities, extracts deployment lessons, and concludes with limitations and future directions.

6.1 The Deployed System

The three AI capabilities presented in Chapters 3–5 are not standalone prototypes. They are integrated into a production platform used daily by bankers, fund screening operators, and tax specialists.

The platform follows a four-layer architecture. The *frontend* is an Angular application providing the conversational interface, document selection, and data visualization panels (Figure 6.1). The *Java backend* (Spring Boot) manages authentication, document metadata, Elasticsearch writes, and serves as the gateway to the bank's existing microservices. The *Python backend* (Flask) hosts the AI logic, the architecture is an orchestrator that routes each user query to the appropriate tool chain, plus the indexation pipeline that processes new documents into the vector store. The *inference layer* consists of an Elasticsearch cluster for vector storage and retrieval, a llama.cpp sidecar for on-premise embedding and reranking, and Azure-hosted models for cloud-eligible workloads.

Tool-routing agent. The Python backend implements a semi-autonomous agent that selects which tools to invoke for each query. The agent sends the query, conversation history, and available tool descriptions to the LLM, which returns a tool selection that is normalized into an executable chain. Six tools are available: *RAG* for document question answering with page citations (Chapter 3), *Summary* for document summarization, *Document Selector* for identifying relevant research documents for a specific portfolio, *Context Loader* for retrieving documents by content classification, *Statistics* for querying structured metadata on a document, and *Data Visualization* for rendering Highcharts charts from statistical results (Figure 6.2). Exclusive tools (RAG, Summary, Document Selector) cannot be combined, while composable tools

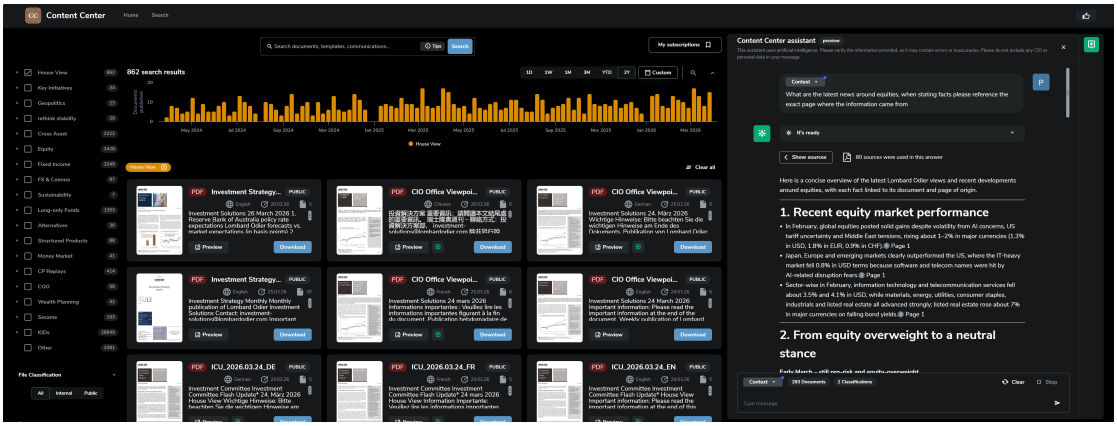

Figure 6.1: Production user interface. The left panel provides document filters by content classification (CIO views, equities, fixed income, cross-asset dailies) and language. Users select one or more categories to query across all matching documents. The right panel shows a multi-document RAG response with page-level citations (e.g., "@ Page 1") linking each claim to its source.

(Context Loader, Statistics, Data Visualization) can be chained. The ALBA advisory pipeline (Chapter 4) and the anomaly detection system (Chapter 5) operate as separate services outside this agent, reflecting their batch interaction patterns.

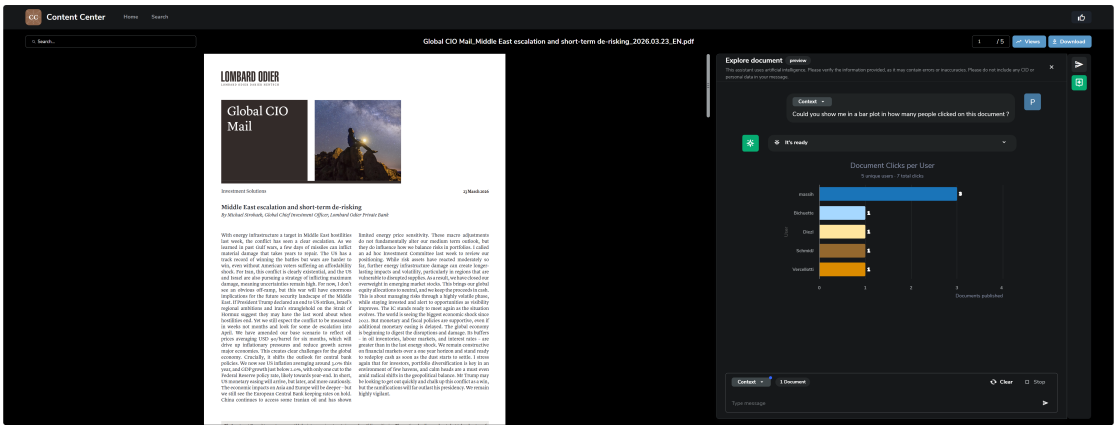

Figure 6.2: Single-document view with data visualization. The left panel displays a CIO publication, while the right panel shows a Highcharts chart generated by the Statistics and Data Visualization tool chain.

6.2 Comparative Analysis of Three AI Modalities

Table 6.1 summarizes the three systems along five dimensions.

The three systems represent increasing levels of autonomy. RAG augments the user, who verifies every answer against the cited pages. ALBA assists the banker, who reviews each

Table 6.1: Comparison of three production AI systems across key design dimensions.

Dimension	RAG (Ch. 3)	Advisory (Ch. 4)	Anomaly (Ch. 5)
AI modality	Retrieval + Generation	Generative reasoning	Statistical learning
Autonomy level	Human-in-the-loop	Human-validated	Autonomous
Latency profile	Interactive (<30 s)	Batch (~4 h CPU)	Batch (~7 s/stmt)
Primary risk	Irrelevant retrieval	Hallucinated trades	False positives
Update cadence	Per document	Per CIO cycle	Annual retrain

recommendation before presenting it to a client. The anomaly detection system operates autonomously across all statements, with errors surfacing only during manual audit. As autonomy increases, failures become harder to detect, which argues for stronger oversight on the more autonomous systems.

A common design principle across all three systems is that task-matched architectures outperform more general alternatives. The RAG system combines a retriever with a generator rather than relying on parametric memory, ALBA uses structured context injection rather than fine-tuning, and the anomaly system uses classical regression rather than neural approaches for structured tabular data. We observed the same principle in separate work on vision-language models, where a task-specific architecture outperformed larger end-to-end models (Massih and Cosatto, 2026).

6.3 Production Deployment Lessons

We organize deployment lessons through the lens of hidden technical debt in ML systems (Sculley et al., 2015).

Pipeline debt. The Java/Python callback architecture (Section 3.2.6) exists because provisioning direct Elasticsearch write access for a new Python codebase required a multi-month security review. The HTTP callback was deployable in days but created a permanent coupling: every schema change requires coordinated releases across both services. In both the RAG and anomaly detection systems, integration glue exceeded core ML logic in engineering effort. Organizations should budget integration at 2–3× the core model development time.

Infrastructure debt. The ALBA pipeline's 4-hour CPU inference time (Section 4.2.3) is the most visible infrastructure constraint. CPU-only LLM inference via llama.cpp is production-viable for embedding and reranking (seconds of latency), but impractical for generative workloads exceeding 100,000 input tokens. The bank has since approved cloud processing for anonymized portfolios, reducing this to minutes.

Monitoring. The anomaly detection system exposes a Kibana dashboard for triaging flagged cases (Figure 6.3). The RAG system is monitored through query logs and user feedback. The advisory pipeline has no production monitoring beyond manual review. Across all three systems, we lack automated drift detection—the single highest-priority engineering improvement.

Figure 6.3: Kibana dashboard for the anomaly detection system. The dashboard shows 243,547 documents processed through the DocGen pipeline and 11,143 flagged as potentially inconsistent. The document count reflects the full pipeline throughput, which includes repeated runs. The deduplicated Swiss client scope comprises 21,679 statements (Section 5.4.3). Confidence codes 90–99 correspond to residual threshold quantiles from Section 5.3.3. Code 102 indicates an equality constraint violation.

Embedding model quality across the optimization path. The indexation optimization in Chapter 3 changed both the inference framework and the embedding model. Table 6.2 confirms that each transition improved embedding quality on the MTEB Multilingual v2 benchmark.[1]

Table 6.2: MTEB Multilingual v2 benchmark comparison of embedding models used across the optimization path.

Model	Params	Dim	Max tokens	Rank	Retrieval	STS
MXBAI-embed-large	335 M	1,024	512	70	40.30	60.46
Qwen3-Embedding-0.6B	596 M	1,024	32,768	13	64.65	76.17
text-embedding-3-large	—	3,072	8,191	32	59.32	71.68

Qwen3-Embedding-0.6B ranks 13th with a retrieval score of 64.65—a 60% improvement over MXBAI's 40.30. The production model, text-embedding-3-large, ranks 32nd but benefits from GPU-accelerated inference and 3,072-dimensional embeddings. The 95.6% indexation speedup was achieved while also improving embedding quality at each stage.

[1] https://huggingface.co/spaces/mteb/leaderboard

Query latency and the 30-second SLO. The RAG system operates under a service-level objective of 30 seconds to get all chunks. This budget is consumed by query contextualization (~2–4 s), embedding (<1 s via Azure), Elasticsearch kNN retrieval (~3–8 s). The primary risk is the optional reranking step: when invoked (Algorithm 1), the 300 reranker forward passes add approximately 1 minute.

6.4 Regulatory Considerations

All three systems operate under FINMA Guidance 08/2024 (Swiss Financial Market Supervisory Authority (FINMA), 2024). Three requirements shaped our architecture. First, *traceability*: the RAG system's page-level citations, ALBA's structured audit trail, and the anomaly system's interpretable ElasticNet coefficients all provide source attribution. Second, *human oversight*: RAG is inherently human-in-the-loop, ALBA recommendations are reviewed by a banker before client presentation, and anomaly flags are reviewed by the tax operations team. Third, *data sovereignty*: ALBA's three-stage architecture processes client portfolios entirely on-premise, while document processing uses cloud models only for content cleared for external processing.

6.5 Limitations

Five limitations affect the interpretation of our results. First, all systems are deployed at a single institution. Performance numbers may not transfer without adaptation. Second, the advisory evaluation spans three structurally diverse test portfolios, and a larger sample would further strengthen generalization claims (Section 4.3.1). Third, anomaly detection precision (80%) is based on 10 escalated cases, and not on the full set of flagged anomalies, as it is difficult to review all flags manually. Fourth, evaluation metrics are automated heuristics that do not capture the full nuance of expert judgment. Fifth, confidentiality constraints prevent disclosure of absolute portfolio values and certain internal identifiers, limiting reproducibility. The RAG benchmark is fully public to partially address this.

6.6 Future Work

Multi-modal RAG. The current pipeline cannot process tables, charts, or images embedded in PDFs. Vision-language models that accept full page images as input would enable a simpler architecture where each PDF page is rendered as an image and interpreted jointly, eliminating the primary quality limitation of text-only extraction.

Advanced query strategies. Hypothetical Document Embeddings (HyDE) (Gao et al., 2023) and query decomposition (Ammann et al., 2025) could address the retrieval failures on easy questions where structurally repeated content dilutes top-k precision (Section 3.3.3).

Frontier model migration for ALBA. The bank's approval of anonymized cloud processing re-

moves the constraint that motivated on-premise generation. Replacing Qwen3-30B-A3B with a frontier model for Stage 3 would reduce inference from hours to minutes while preserving the same architecture.

Temporal features for anomaly detection. Adding holding period information would allow the model to distinguish partial year positions from genuine errors, addressing both the regression based and count based false positives (Section 5.5).

Automated drift detection. Implementing retrieval quality regression tests, advisory consistency checks, and anomaly calibration monitoring would close the monitoring gap identified in Section 6.3. The 20-question RAG benchmark (Section 3.3.3) can serve as a regression test by re-running it after each model or configuration change.

6.7 Concluding Remarks

This thesis presented the design, implementation, and production deployment of three AI systems at Lombard Odier, each targeting a distinct operational bottleneck.

The RAG architecture (Chapter 3) reduced document indexation time by 95.6%, achieved 95% page citation accuracy on a 20-question French-language benchmark, and demonstrated that imatrix-calibrated K-quant compression reduces embedding model size by 70% with negligible quality loss. The ALBA advisory pipeline (Chapter 4) showed that expert-curated context injection enables a 3B-active-parameter model to outperform GPT-5.1 on domain-specific advisory quality (96% vs. 47% overall), demonstrating that context quality dominates model capability for structured financial reasoning. The anomaly detection system (Chapter 5) reduced the false positive rate from 90% to 5% through domain-informed preprocessing, confirming that in production anomaly detection, data engineering outweighs model architecture.

Across all three deployments, the engineering challenges of production ML in a regulated environment shaped architecture decisions as much as the underlying methodology. The systems described in this thesis demonstrate that production AI in financial services is achievable with current technology, provided that the engineering effort matches the ambition.

A PrivilEdge RAG Benchmark

Table A.1 lists all 20 questions in the PrivilEdge Prospectus RAG Benchmark, with ground-truth answers and source pages. The dataset is publicly available at https://huggingface.co/datasets/PeterAM4/priviledge-prospectus-rag-benchmark.

Table A.1: Full PrivilEdge RAG Benchmark (20 questions). All questions and answers are in French, targeting the 354-page PrivilEdge prospectus (Feb 2026).

#	Question	Ground-Truth Answer	Page	Diff.	Category
1	Quel est le montant minimum de souscription et de détention pour les Actions de classe U?	L'équivalent d'EUR 25'000'000.	78	E	Fee
2	Qui est le Représentant en Suisse de PrivilEdge et à quelle adresse?	Lombard Odier Asset Management (Switzerland) SA, Parc Messidor 3, 1293 Bellevue.	3	E	Gov.
3	Commission de souscription maximale pour les Actions de classe R?	Ne doit pas dépasser 3% du Prix d'émission.	50	E	Fee
4	Organe de révision de PrivilEdge?	PricewaterhouseCoopers Assurance, 2 rue Gerhard Mercator, 1471 Luxembourg.	50	E	Gov.
5	Décimales d'arrondi de la VNI par Action, et exception?	Quatre décimales, sauf Actions libellées en JPY.	63	E	Fund
6	Limite maximale d'investissement en valeurs mobilières d'un même émetteur (§4.2), et cas de 35%?	10% des actifs nets; 35% si émises/garanties par un État Membre ou organisme international.	37	M	Restr.
7	Conditions pour investir jusqu'à 100% dans des valeurs d'un État?	Six émissions différentes min., max. 30% par émission. États OCDE, G20, Singapour, UE.	37	M	Restr.

Continued on next page

Table A.1: (continued)

#	Question	Ground-Truth Answer	Page	Diff.	Category
8	Commission de performance pour Polar Capital Global Healthcare?	10% de la surperformance vs MSCI World Health Care. Cristallisation: 31 déc.	85	M	Fee
9	Formule de conversion des Actions entre Compartiments?	$A = (B \times C \times D - F)/E$, avec commission jusqu'à 0,50%.	60	M	Fund
10	Seuil de rachat pour ajournement et durée maximale?	10% des Actions; max. 7 Jours d'évaluation.	59	M	Risk
11	Taux de la taxe d'abonnement pour classe S (institutionnels) et taux standard?	0,01% (classe S); 0,05% (standard).	66	M	Tax
12	Levier attendu, fréquence d'évaluation et jour pour Graham Quant Macro?	800% VNI (somme notionnels). Hebdomadaire, mercredi.	173	H	Fund
13	Comparer Commission de gestion classe P: Mondrian US Equity Value vs Allianz All China Equity?	Mondrian: 0,50%; Allianz: 0,85%. Mondrian plus basse.	99	H	Fee
14	Administrateur de l'Indice de référence de PPM America US Corporate Bond, inscrit AEMF?	ICE Data Indices, LLC. N'apparaît pas dans le registre AEMF.	159	H	Bench.
15	Compartiments autorisés à effectuer des ventes à découvert de dérivés?	Non (§4.2(m)).	39	H	Restr.
16	Commission de performance pour JPMorgan US Equities si VNI=120 et indice=101,87?	2,19% (15%×14,59%); valeur absolue 2,63.	93	VH	Fee
17	Délai d'abandon du produit de liquidation non réclamé et lieu de dépôt?	30 ans; Caisse de Consignation du Luxembourg.	65	VH	Fund
18	Compartiment à évaluation hebdomadaire? FROC max. le plus bas en classe S?	Graham Quant Macro (mercredi). FROC 0,10%: Robeco EM Equities & BlueBay IG Govt Bonds.	132	VH	Fund
19	% min. de VNI qu'un Nourricier doit investir dans un Maître? Possible en Suisse?	85% min. Aucun Compartiment distribué en Suisse ne se qualifie.	38	VH	Restr.
20	Exposition attendue de Graham Quant Macro aux TRS (somme des notionnels)?	200%–300% VNI; si dépassement, reste <500%.	177	VH	Risk

B Full GGUF Quantization Results

Table B.1 reports perplexity for all 30 GGUF quantization variants of Qwen3-Embedding-0.6B evaluated on the 22 MB financial calibration corpus (context window 1,536 tokens, 200 chunks). The BF16 baseline achieves a perplexity of 406.03.

Table B.1: Complete GGUF quantization ablation. Δ PPL is relative to BF16 baseline (406.03).

Model	Family	BPW	Size	Δ PPL
BF16 (baseline)	–	16.08	1.1 GB	–
Q8_0	Legacy	8.58	610 MB	+3.54
Q6_K	K-quant	6.65	472 MB	+11.35
Q5_1	Legacy	6.23	442 MB	+20.92
Q5_K_M	K-quant	5.96	424 MB	+36.92
Q5_0	Legacy	5.86	416 MB	+7.17
Q5_K_S	K-quant	5.86	416 MB	+8.91
Q4_1-imat	Legacy	5.49	390 MB	−2.96
Q4_K_M-imat	K-quant	5.32	378 MB	+0.65
Q4_K_S-imat	K-quant	5.14	365 MB	+0.97
Q4_0-imat	Legacy	5.13	364 MB	+13.86
IQ4_NL-imat	I-quant	5.12	364 MB	+29.00
Q3_K_L-imat	K-quant	4.94	351 MB	+6.00
IQ4_XS-imat	I-quant	4.94	351 MB	+45.38
Q3_K_M-imat	**K-quant**	**4.66**	**331 MB**	**+0.62**
IQ3_M-imat	I-quant	4.51	320 MB	+54.92
IQ3_S-imat	I-quant	4.34	308 MB	+69.45
Q3_K_S-imat*	K-quant	4.34	308 MB	−65.73
IQ3_XS-imat	I-quant	4.20	298 MB	+114.37
Q2_K-imat	K-quant	3.97	282 MB	+391.83
Q2_K_S-imat	K-quant	3.76	267 MB	+1155.22
IQ3_XXS-imat	I-quant	3.74	266 MB	+207.91
IQ2_M-imat	I-quant	3.55	252 MB	+877.42
IQ2_S-imat	I-quant	3.41	242 MB	+1451.39
TQ2_0-imat	Ternary	3.32	236 MB	diverged
IQ2_XS-imat	I-quant	3.25	231 MB	+3226.90
IQ2_XXS-imat	I-quant	3.08	219 MB	+5235.87
TQ1_0-imat	Ternary	3.04	216 MB	diverged
IQ1_M-imat	I-quant	2.90	206 MB	+7089.39
IQ1_S-imat	I-quant	2.79	198 MB	+23008.92

*Statistical artifact; not a genuine improvement over baseline.

C ALBA Advisory Output Comparison

This appendix presents the full output for one investment idea from each condition to illustrate the qualitative difference between a frontier model with web search (Condition A) and the ALBA pipeline (Condition C). Both recommend Taiwan Semiconductor Manufacturing Co. (TSMC), allowing a direct comparison.

Condition A: GPT-5.1 with Web Search

Field	**Value**
Title	Rebuild EM via Taiwan semiconductor champion TSMC
Buy	Taiwan Semiconductor Manufacturing Company Limited ADR
ISIN	US8740391003 (US-listed ADR)
Bucket	Eq.EM
Suggested weight	1.8%
Research rating	*Not available (no access to internal research)*
Price target	*Not available*
Funding	
Sell 1	Deutsche Bank (DE0005140008), trim 0.8% from 4.72%
Sell 2	ENI (IT0003132476), trim 0.5% from 3.67%
Sell 3	BP (GB0007980591), trim 0.5% from 1.07%

Condition A correctly identifies the zero emerging-market exposure and recommends TSMC. However, it uses the US-listed ADR (ISIN US8740391003) rather than the Taiwan-listed shares (TW0002330008) that the bank rates. The recommendation includes no research rating, no price target, and no valuation multiples. The sells reference real positions with approximately correct weights.

Field	Value
Title	Add TSMC: core EM and AI exposure in line with CIO EM overweight
Buy	Taiwan Semiconductor Manufacturing Co. Ltd.
ISIN	TW0002330008 (Taiwan-listed)
Bucket	Eq.EM
Suggested weight	2.0%
Research rating	Strong Buy
Price target	TWD 2,300 (upside: 18.9%)
P/E (2025E / 2026E)	29.2x / 22.5x
Dividend yield	1.1%
Funding	
Sell 1	Deutsche Bank (DE0005140008), trim 1.0% from 4.27% to 3.27%
Sell 2	USD fiduciary on-call deposit, trim 1.0% from 4.1% to 3.1%
CIO alignment	
Adds emerging-market Asia and AI infrastructure exposure in line with the CIO's EM equity overweight and preference for semiconductor and AI beneficiaries.	

Condition C: ALBA

Banker-facing rationale (excerpt)."CIO/TAA gap: Eq.EM is 0% vs 7.1% TAA target, while the CIO is firmly overweight emerging-market equities, especially Asia. TSMC is the key global foundry for advanced chips and a central beneficiary of AI, cloud and edge-computing demand. It is rated Strong Buy by our research with attractive upside and superior earnings visibility. Funding: trim Deutsche Bank (large Eurozone bank overweight, rated Hold) and a portion of the USD fiduciary on-call deposit."

Client-facing text (excerpt)."You currently have no emerging-market equities, even though we see attractive opportunities there, especially in Asia. TSMC is at the heart of the AI and digitalisation boom, and its chips are used in data centres, cloud computing and many connected devices. Adding TSMC gives you exposure to Asian growth and the AI theme, while we slightly reduce your exposure to Eurozone banks and cash-like instruments that are already very large in your portfolio."

Key differences. Three concrete differences. First, ALBA uses the correct ISIN (Taiwan-listed TW0002330008) that the bank trades and rates, while GPT-5.1 uses the US ADR requiring a different settlement process. Second, ALBA provides the bank's own research rating, price target, and valuation multiples, making the recommendation immediately presentable to a client. Third, ALBA's rationale explicitly references the CIO's EM overweight commandment and the portfolio's 0% vs 7.1% TAA deviation, grounding the recommendation in the bank's investment framework rather than generic market commentary.

D ALBA Master Prompt

This appendix reproduces the master prompt used in Stage 3 of the ALBA pipeline (Section 4.2.3). The prompt is sent to Qwen3-30B-A3B alongside the assembled structured context (CIO views, sector comments, equity research list, enriched portfolio, and trade history). The JSON output schema is omitted for brevity.

```
MASTER PROMPT - PORTFOLIO REVIEW
(MACRO → MICRO, CLIENT-FACING POSITIVE TONE, STRICT JSON)

1) Role and objective
You are an Investment Adviser at a private bank.
Your task: use CIO/IC research to identify macro opportunities,
then use the bank's actionable ideas as vehicles to express
those views in the portfolio. Propose at most ONE buy trade
sized around "Average Positions Weight", and sells only if
needed to free cash. Think: Macro first, Structure second,
Instruments third.

2) Inputs you will receive
a) Enriched portfolio JSON with precomputed metrics:
   clientNo, baseCurrency, totalPortfolioValue,
   averagePositionWeightPct, cash statistics,
   exposuresByBucket (target/actual/deviation per TAA bucket),
   exposuresByRiskCurrency, largestOverweights/Underweights,
   full positions array.
b) CIO/IC materials, Investment Ideas with ratings, optional
   thematic notes.

3) Strict hierarchy: Macro → Structure → Instruments
   1. Macro: identify CIO preferences, define 1-2 themes.
   2. Structure: use exposuresByBucket to find misalignments.
   3. Instruments: select from research list to implement themes.
```

```
4) Use precomputed metrics (do not recalculate)
   Trust totalPortfolioValue, exposures, deviations.

5) Attribution discipline
   Only claim a view as CIO/IC if explicitly stated.
   Never say CIO "emphasised" unless the text does so.

6) Cash and sell rules
   If cashSufficientForAPWBuy: NO sells, fund from cash.
   Otherwise: sell-rated instruments first, then overweights.

7) Sizing: default = averagePositionWeightPct, adjust for
   conviction and bucket deviations.

8) Client-facing tone: professional, positive, no "AI"/"model"/
   "bucket"/"TAA". Buy justification 3-6 sentences,
   sell justification 2-5 sentences.

9) Output: strict JSON with clientNo, thinkSummary, buyTrade
   (or null), sellTrades (array or null).
```

The production version extends this base prompt to generate multiple investment ideas organized across five strategic axes (CIO/TAA alignment, strategic themes, micro opportunities, relative-value switches, and portfolio risk), each with dual-tone messaging (banker-facing and client-facing). The core reasoning hierarchy and constraint structure remain identical.

Bibliography

Ammann, P. J. L., Golde, J., and Akbik, A. (2025). Question decomposition for retrieval-augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pages 497–507, Vienna, Austria. Association for Computational Linguistics.

Araci, D. (2019). FinBERT: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063*.

Bhat, S. R., Rudat, M., Spiekermann, J., and Flores-Herr, N. (2025). Rethinking chunk size for long-document retrieval: A multi-dataset analysis. *arXiv preprint arXiv:2505.21700*.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., et al. (2020). Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33.

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):1–58.

Chen, J., Xiao, S., Zhang, P., Luo, K., Lian, D., and Liu, Z. (2024). BGE M3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*.

De Roux, D., Pérez, B., Moreno, A., del Pilar Villamil, M., and Figueroa, C. (2018). Tax fraud detection for under-reporting declarations using an unsupervised machine learning approach. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 215–222.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 4171–4186.

Elastic NV (2024). Elasticsearch: Distributed search and analytics engine. https://www.elastic.co/elasticsearch.

Fatouros, G., Metaxas, K., Soldatos, J., and Kyriazis, D. (2025). Can large language models beat wall street? Evaluating GPT-4's impact on financial decision-making with MarketSenseAI. *Neural Computing and Applications*, 37:24893–24918.

Fieberg, C., Hornuf, L., Meiler, M., and Streich, D. (2025). Using large language models for financial advice. CESifo Working Paper 11666, CESifo.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. (2023). GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations (ICLR)*.

Gao, L., Ma, X., Lin, J., and Callan, J. (2023). Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1762–1777. Association for Computational Linguistics.

Gao, Y., Xiong, Y., Vejber, S., Lin, T., et al. (2024). Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Gerganov, G. (2023). llama.cpp. https://github.com/ggerganov/llama.cpp.

GraphQL Foundation (2025). Graphql specification. https://spec.graphql.org/September2025/. Latest release, accessed March 27, 2026.

Hoerl, A. E. and Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

Karakurt, E. and Akbulut, A. (2026). Retrieval-augmented generation (RAG) and large language models (LLMs) for enterprise knowledge management and document automation: A systematic literature review. *Applied Sciences*, 16(1):368.

Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. (2020). Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 6769–6781.

Kreps, J., Narkhede, N., and Rao, J. (2011). Kafka: A distributed messaging system for log processing. In *Proceedings of the NetDB Workshop*.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Li, B., Gu, B., and Ding, Z. (2025). LLM-based personalized portfolio recommender. *arXiv preprint arXiv:2512.12922*.

Lin, J., Tang, J., Tang, H., Yang, S., Chen, W.-M., Wang, W.-C., Xiao, G., Dang, X., Gan, C., and Han, S. (2024). AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *Proceedings of Machine Learning and Systems (MLSys)*.

LlamaIndex (2024). LlamaIndex: Data framework for LLM applications. https://github.com/run-llama/llama_index.

Malkov, Y. A. and Yashunin, D. A. (2020). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836.

Massih, P. and Cosatto, E. (2026). Reasoning with pixel-level precision: QVLM architecture and SQuID dataset for quantitative geospatial analytics. *arXiv preprint arXiv:2601.13401*.

Mazumder, R., Hastie, T., and Tibshirani, R. (2010). Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research*, 11:2287–2322.

McKinsey Global Institute (2012). The social economy: Unlocking value and productivity through social technologies. Technical report, McKinsey & Company.

Muennighoff, N., Tazi, N., Magne, L., and Reimers, N. (2023). MTEB: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*.

n8n GmbH (2025). n8n: Fair-code workflow automation platform. https://github.com/n8n-io/n8n. Version 1.x.

ONNX Runtime developers (2021). ONNX runtime. https://onnxruntime.ai/. Version 1.17.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training. *OpenAI Technical Report*.

Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Robertson, S. and Zaragoza, H. (2009). The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

Sculley, D., Holt, G., Golovin, D., Ebner, D., et al. (2015). Hidden technical debt in machine learning systems. In *Advances in Neural Information Processing Systems*, volume 28.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations (ICLR)*.

Singer-Vine, J. (2024). pdfplumber: Plumb a PDF for detailed information about each text character, rectangle, and line. https://github.com/jsvine/pdfplumber. Accessed: 2026-03-27.

Swiss Federal Tax Administration (ESTV/AFC) (2024). ICTax: Cours et dividendes des valeurs mobilières suisses et étrangères. https://www.ictax.admin.ch.

Swiss Financial Market Supervisory Authority (FINMA) (2024). Guidance 08/2024: Artificial intelligence — governance and risk management. https://www.finma.ch/en/~/media/finma/dokumente/dokumentencenter/myfinma/4dokumentation/finma-aufsichtsmitteilungen/20241205-finma-aufsichtsmitteilung-08-2024.pdf.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B*, 58(1):267–288.

Vanhoeyveld, J., Martens, D., and Peeters, B. (2020). Value-added tax fraud detection with scalable anomaly detection techniques. *Applied Soft Computing*, 86:105895.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.

Wu, S., Irsoy, O., Lu, S., Dabravolski, V., Dredze, M., Gehrmann, S., Kambadur, P., Rosenberg, D., and Mann, G. (2023). BloombergGPT: A large language model for finance. *arXiv preprint arXiv:2303.17564*.

Xiao, S., Liu, Z., Zhang, P., and Muennighoff, N. (2023). C-pack: Packaged resources to advance general chinese embedding. *arXiv preprint arXiv:2309.07597*.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., et al. (2025). Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Yang, H., Liu, X.-Y., and Wang, C. D. (2023). FinGPT: Open-source financial large language models. *arXiv preprint arXiv:2306.06031*.

Zhang, Y., Li, M., Long, D., Zhang, X., Lin, H., Yang, B., Xie, P., Yang, A., Liu, D., Lin, J., Huang, F., and Zhou, J. (2025). Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B*, 67(2):301–320.