LLM-Powered Customer Churn Analysis with Retrieval-Augmented Generation and QLoRA Fine-Tuning

Ricky Gong
gong8@seas.upenn.edu

Abstract

Customer churn prediction and analysis remain critical challenges in the telecommunications industry, where the average monthly churn rate of 1.5–2% translates to an estimated $65 billion in annual revenue loss in the United States alone. Traditional machine learning approaches to churn prediction focus on binary classification but fail to provide the interpretable, actionable analysis that business stakeholders require. In this paper, we present CHURN-RAG, an end-to-end system that combines Retrieval-Augmented Generation (RAG) with QLoRA fine-tuning of open-source large language models for comprehensive customer churn analysis. Our system employs a hybrid retrieval pipeline combining dense vector search with BM25 keyword matching via Reciprocal Rank Fusion, paired with the Qwen2.5-7B-Instruct model quantized to 4-bit NormalFloat (NF4) precision. We introduce a teacher-student distillation pipeline using Qwen2.5-14B-Instruct to generate 305 high-quality domain-specific training samples, enabling QLoRA fine-tuning on a single consumer GPU. To address post-fine-tuning regressions in citation accuracy and risk assessment consistency, we propose a hybrid inference pipeline that augments LLM generation with deterministic post-processing for citation validation and mathematical risk scoring. Our improved pipeline achieves 100% JSON format compliance, 100% citation accuracy, and consistent risk level alignment across all evaluation queries, demonstrating that targeted post-processing can complement fine-tuning to produce reliable, production-ready outputs without additional retraining.

1 Introduction

1.1 Industry Background

The telecommunications industry faces a persistent and costly challenge: customer churn. With average monthly churn rates of 1.5–2% across major carriers, the cumulative financial impact is substantial—estimated at $65 billion in annual revenue loss in the United States alone [8]. The cost of acquiring a new customer is five to seven times higher than retaining an existing one, making churn prevention one of the highest-ROI activities for telecom operators.

Traditional approaches to churn analysis have relied primarily on supervised machine learning models—logistic regression, random forests, gradient-boosted trees, and more recently deep neural networks—trained to predict a binary churn/no-churn outcome [12]. While these models can achieve reasonable predictive accuracy, they suffer from a fundamental limitation: they tell *whether* a customer will churn, but not *why*, nor what *specific actions* should be taken to prevent it. Business stakeholders require not just predictions but interpretable, contextual analyses grounded in actual customer data.

1.2 Problem Statement

We seek to build a scalable, interpretable churn analysis system that, given a natural language query (e.g., "Why are fiber optic customers churning?"), retrieves relevant customer records and generates a structured analysis report containing: (1) a summary of key findings, (2) the top reasons for churn, (3) a calibrated risk level, (4) specific recommended actions, and (5) citations to the customer records that support the analysis.

1.3 Project Context

This project was adapted from a customer churn analysis system originally developed during an internship at Eth Tech, a technology company. Due to data compliance restrictions and budget constraints, we made the following adaptations for the open-source implementation:

- **Dataset**: We use the publicly available Kaggle "Telco Customer Churn with Realistic Feedback" dataset [6] instead of proprietary customer data.
- **Models**: We use the open-source Qwen2.5 model family [10] instead of proprietary APIs, enabling fully reproducible, zero-cost execution on Google Colab.

1.4 Dataset Description

The Kaggle Telco Customer Churn dataset contains 7,043 customer records with 21 structured features covering demographics (gender, senior citizen status, partner, dependents), account information (tenure, contract type, payment method, monthly/total charges), service subscriptions (phone, internet, online security, tech support, streaming), and churn status. Crucially, the dataset also includes a free-text `CustomerFeedback` field for each customer, providing natural language descriptions of their experience. This combination of structured and unstructured data makes the dataset well-suited for RAG-based analysis.

1.5 Technical Approach

Our system, ChurnRAG, integrates three key technologies:

1. **Retrieval-Augmented Generation (RAG)** [9]: Rather than relying solely on the LLM's parametric knowledge, RAG retrieves the most relevant customer records for each query and includes them in the prompt context. This addresses context window limitations (7,043 documents at $\sim$300–500 tokens each far exceed even 128K context windows), ensures verifiability (every claim can be traced to a specific customer record), and reduces hallucination.
2. **QLoRA Fine-Tuning** [3]: While general-purpose LLMs can follow structured output instructions, they lack domain-specific knowledge about telecom churn patterns. QLoRA enables parameter-efficient fine-tuning of a 7-billion-parameter model on a single consumer GPU by combining 4-bit NormalFloat quantization with Low-Rank Adaptation [4], training only 0.53% of the model's parameters.
3. **Deterministic Post-Processing**: Fine-tuning can introduce regressions in specific capabilities. We augment LLM generation with rule-based citation validation and a mathematical risk scoring model, combining the benefits of neural generation with the reliability of deterministic algorithms.

1.6 Contributions

This paper makes the following contributions:

1. An end-to-end RAG + QLoRA system for customer churn analysis that generates structured, citation-backed analysis reports from natural language queries.
2. A teacher-student distillation pipeline that uses a larger 14B-parameter model to generate high-quality domain-specific training data for fine-tuning a 7B-parameter student model.
3. A hybrid inference pipeline that combines LLM generation with deterministic post-processing to fix citation accuracy regression and risk assessment instability without retraining.

2 Related Work

Retrieval-Augmented Generation. RAG was introduced by Lewis et al. [9] as a method to combine parametric (neural) and non-parametric (retrieval) memory for knowledge-intensive NLP tasks. Subsequent work has explored dense retrieval [7], hybrid retrieval combining dense and sparse methods, and the application of RAG to domain-specific tasks such as question answering, fact verification, and enterprise search. Our work applies RAG to customer analytics, where the "knowledge base" consists of structured customer records augmented with free-text feedback.

Parameter-Efficient Fine-Tuning. LoRA (Low-Rank Adaptation) [4] freezes pretrained weights and injects trainable low-rank decomposition matrices into transformer layers, dramatically reducing the number of trainable parameters. QLoRA [3] extends this by quantizing the frozen weights to 4-bit NormalFloat precision, enabling fine-tuning of models with billions of parameters on consumer GPUs. Our work applies QLoRA to domain-specific fine-tuning for structured output generation in the customer analytics domain.

LLMs for Business Analytics. Recent work has explored using large language models for business intelligence tasks including sentiment analysis, customer feedback summarization, and report generation. However, most approaches rely on proprietary APIs (GPT-4, Claude) with associated cost and data privacy concerns. Our approach uses fully open-source models, enabling deployment in privacy-sensitive enterprise environments.

Customer Churn Prediction. The customer churn prediction literature is extensive, spanning logistic regression [12], random forests, XGBoost, and deep learning approaches. However, these methods focus on binary prediction rather than interpretable analysis. Our work shifts from prediction to analysis: given that churn has occurred (or is likely), we explain why and recommend specific interventions.

3 Methodology

3.1 Data Preparation

Dataset Statistics. Table 1 summarizes the key statistics of the Kaggle Telco Customer Churn dataset.

Table 1: Dataset statistics.

Property	Value
Total customers	7,043
Structured features	21
Churn rate	26.5% (1,869 churned)
Tenure range	0–72 months
Monthly charges range	\$18.25–\$118.75
Contract types	Month-to-month, One year, Two year
Internet service types	DSL, Fiber optic, None
Customers with feedback	7,043 (100%)

Document Construction. Each customer record is converted into a unified text document that combines structured profile data with the free-text feedback field. A document follows the template:

```
Customer ID: {id}
Churn Status: {Yes/No}
Customer Profile: Gender, Senior Citizen, Partner, ...
Services: Phone, Internet, Security, ...
Contract & Billing: Contract type, Charges, Payment, ...
Customer Feedback: {free-text feedback}
```

This unified representation enables both semantic and keyword-based retrieval over both structured and unstructured fields simultaneously.

3.2 Retrieval System

Our retrieval pipeline combines two complementary methods—dense vector search for semantic similarity and BM25 for keyword matching—fused via Reciprocal Rank Fusion.

3.2.1 Vector Search

Vector search identifies semantically similar documents by computing cosine similarity between query and document embeddings. Given a query embedding $\mathbf{q} \in \mathbb{R}^d$ and a document embedding $\mathbf{d}_i \in \mathbb{R}^d$, the cosine similarity is:

$$\text{sim}(\mathbf{q}, \mathbf{d}_i) = \frac{\mathbf{q} \cdot \mathbf{d}_i}{\|\mathbf{q}\| \cdot \|\mathbf{d}_i\|} \tag{1}$$

We use `BAAI/bge-base-en-v1.5` [13] as the embedding model, which produces 768-dimensional L2-normalized vectors. Since all vectors are unit-normalized, cosine similarity reduces to the dot product:

$$\text{sim}(\mathbf{q}, \mathbf{d}_i) = \mathbf{q} \cdot \mathbf{d}_i \tag{2}$$

Document embeddings are indexed using FAISS [5] with `IndexFlatIP` (inner product), enabling exact nearest-neighbor search over the full corpus of 7,043 documents.

3.2.2 Keyword Search (BM25)

BM25 (Best Matching 25) [11] complements vector search by matching exact keywords and term frequencies. For a query Q consisting of terms $q_1, q_2, \ldots, q_n$, the BM25 score for document D is:

$$\text{BM25}(D, Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)} \tag{3}$$

where:

- $\text{IDF}(q_i) = \ln\left(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} + 1\right)$ is the inverse document frequency of term q_i,
- N is the total number of documents in the corpus (7,043),
- $n(q_i)$ is the number of documents containing term q_i,
- $f(q_i, D)$ is the frequency of term q_i in document D (term count),
- $|D|$ is the length of document D in tokens,
- avgdl is the average document length across the corpus,
- $k_1 = 1.5$ controls term frequency saturation, and
- $b = 0.75$ controls document length normalization.

The combination of vector and keyword search is motivated by their complementary strengths: vector search captures semantic similarity (e.g., "unhappy" matches "dissatisfied") but may miss exact terms (e.g., "Fiber optic"), while BM25 excels at exact keyword matching but cannot capture synonyms or paraphrases.

3.2.3 Reciprocal Rank Fusion (RRF)

To combine the ranked lists from vector search and BM25 into a single ranking, we use Reciprocal Rank Fusion [1]. For a document d appearing in the result lists of retrieval methods $\mathcal{R} = \{\text{vector}, \text{BM25}\}$, the fused score is:

$$\text{RRF}(d) = \sum_{r \in \mathcal{R}} \frac{w_r}{k + \text{rank}_r(d)} \tag{4}$$

where:

- $\text{rank}_r(d)$ is the 0-indexed rank of document d in the result list of method r,
- w_r is the weight assigned to method r (we use $w_{\text{vector}} = \alpha = 0.5$ and $w_{\text{BM25}} = 1 - \alpha = 0.5$ for equal weighting), and
- $k = 60$ is a smoothing constant that reduces the influence of high-ranking outliers.

RRF is particularly well-suited for combining heterogeneous retrieval methods because it operates on *rank positions* rather than raw scores, which may have incomparable scales across different methods. The top-K documents (default $K = 5$) by RRF score form the retrieval context for the LLM.

Retrieval Hyperparameters. Table 2 summarizes the retrieval system configuration.

3.3 LLM Integration

We use `Qwen2.5-7B-Instruct` [10] as the base language model for analysis generation. The model is loaded in 4-bit NF4 quantization for memory-efficient inference (∼3.5 GB GPU memory for model weights).

Table 2: Retrieval system hyperparameters.

Component	Parameter	Value
Vector Search	Embedding model	`BAAI/bge-base-en-v1.5`
	Embedding dimension	768
	Index type	FAISS `IndexFlatIP`
BM25	k_1	1.5
	b	0.75
RRF	k (smoothing)	60
	α (vector weight)	0.5
	Top-K	5

Prompt Design. The LLM receives a two-part prompt: a *system prompt* defining its role as a customer churn analysis expert and specifying the required JSON output schema, and a *user prompt* containing the natural language query and the retrieved customer documents. The required output schema consists of five fields:

1. `summary` (string): A 2–3 sentence overview of the analysis.
2. `top_reasons` (list of strings): The top 3 reasons for churn.
3. `risk_level` (string): One of "high", "medium", or "low".
4. `actions` (list of strings): 3 specific recommended actions.
5. `citations` (list of strings): Customer IDs supporting the analysis.

Inference Configuration. Table 3 lists the inference hyperparameters used during generation.

Table 3: Inference configuration.

Parameter	Value
Model	`Qwen/Qwen2.5-7B-Instruct`
Quantization	4-bit NF4, double quantization
Compute dtype	`bfloat16`
Temperature	0.7
Top-p	0.9
Max new tokens	1,024
Sampling	Enabled (`do_sample=True`)

3.4 Teacher-Student Distillation

To generate high-quality domain-specific training data for fine-tuning the 7B student model, we employ a teacher-student distillation approach using a larger model as the teacher.

Teacher Model. We use `Qwen2.5-14B-Instruct` quantized to 4-bit NF4 ($\sim$8 GB VRAM) as the teacher model. Despite quantization, the 14B model produces substantially higher-quality structured analyses than the 7B model, particularly in terms of reasoning depth and consistency.

Query Generation. We design 8 categories of query templates covering the key dimensions of churn analysis:

1. **Service** (28 queries): Service-specific churn causes (e.g., "Why are customers with fiber optic internet churning?")
2. **Action** (28 queries): Actionable recommendations (e.g., "What should we do to reduce churn among month-to-month customers?")
3. **Demographics** (20 queries): Demographic factor analysis

4. **Tenure** (15 queries): Tenure-based patterns
5. **Comparison** (12 queries): Cross-group comparisons
6. **Contract** (9 queries): Contract-type analysis
7. **Pricing** (5 queries): Pricing and billing queries
8. **Sentiment** (5 queries): Customer sentiment analysis

Through parameterized template expansion (substituting service names, customer groups, contract types, etc.) and deduplication, we generate 122 unique queries.

Data Augmentation. Each query is paired with three different retrieval contexts using $k \in \{3, 5, 7\}$ retrieved documents. This produces diverse training contexts: $k = 3$ teaches the model to work with minimal context, $k = 5$ matches the standard inference setting, and $k = 7$ teaches the model to handle noisy or redundant information. The teacher generates a JSON response for each (query, context) pair, yielding a maximum of $122 \times 3 = 366$ samples. After filtering failed generations (1.9% failure rate), we obtain 359 valid samples.

Data Splitting. The valid samples are split 85%/15% using stratified sampling by query category, yielding **305 training samples** and **54 validation samples**. Each sample is formatted in ChatML format with system, user, and assistant messages.

3.5 QLoRA Fine-Tuning

QLoRA [3] combines two techniques—NF4 quantization and LoRA—to enable fine-tuning of large language models on consumer GPUs.

3.5.1 NF4 Quantization

QLoRA uses NormalFloat 4-bit (NF4) quantization, which is information-theoretically optimal for normally distributed weights. Neural network weights are approximately normally distributed, so the 16 quantization levels are mapped to the quantiles of the standard normal distribution $\mathcal{N}(0, 1)$:

$$q_i = \Phi^{-1}\left(\frac{i + 0.5}{16}\right), \quad i = 0, 1, \ldots, 15 \tag{5}$$

where:
- Φ^{-1} is the quantile function (inverse CDF) of the standard normal distribution $\mathcal{N}(0, 1)$,
- q_i is the i-th quantization level, and
- $i \in \{0, 1, \ldots, 15\}$ indexes the 16 levels for 4-bit quantization.

Each weight w is quantized to the nearest NF4 level: $\hat{w} = \sigma \cdot q_{i^*}$, where σ is a per-block scaling factor (block size = 64 weights) and $i^* = \arg\min_i |w/\sigma - q_i|$.

Double Quantization. The per-block scaling factors σ are themselves quantized to 8-bit precision, yielding additional memory savings. This "quantization of the quantization constants" is a key innovation of QLoRA.

Memory Savings. The 7-billion-parameter model requires approximately 14 GB in FP16 precision. NF4 quantization reduces this to approximately 3.5 GB—a 4× compression with minimal quality degradation due to the information-theoretically optimal quantization grid.

Table 4 lists the quantization configuration.

3.5.2 LoRA Decomposition

Instead of updating the full weight matrix $W_0 \in \mathbb{R}^{d \times k}$, LoRA [4] introduces a low-rank additive update:

$$W' = W_0 + \Delta W = W_0 + BA \tag{6}$$

where:
- $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$ are the trainable low-rank matrices,

Table 4: BitsAndBytesConfig for 4-bit quantization.

Parameter	**Value**
`load_in_4bit`	`True`
`bnb_4bit_quant_type`	`nf4`
`bnb_4bit_compute_dtype`	`bfloat16`
`bnb_4bit_use_double_quant`	`True`

- $r \ll \min(d, k)$ is the rank (we use $r = 16$), and
- W_0 is frozen (receives no gradient updates).

Parameter Savings. For a single linear layer with $d = k = 4096$ (typical for Qwen2.5-7B attention projections):
- Full fine-tuning: $d \times k =$ 16,777,216 parameters.
- LoRA ($r = 16$): $d \times r + r \times k = 2 \times 4096 \times 16 =$ 131,072 parameters.

This represents a $128\times$ reduction per layer.

Forward Pass. During inference, the modified forward pass becomes:

$$h = W_0 x + \frac{\alpha}{r} \cdot BAx \tag{7}$$

where:
- $x \in \mathbb{R}^k$ is the input activation vector,
- $h \in \mathbb{R}^d$ is the output activation vector,
- $\alpha = 32$ is the scaling factor that controls the magnitude of the LoRA update, and
- $r = 16$ is the rank.

LoRA Configuration. Table 5 shows our LoRA configuration, which targets all attention and MLP projection layers for maximum coverage.

Table 5: LoRA adapter configuration (from `adapter_config.json`).

Parameter	**Value**
Rank (r)	16
Alpha (α)	32
Dropout	0.05
Bias	None
Task type	`CAUSAL_LM`
Target modules	`q_proj`, `k_proj`, `v_proj`, `o_proj` `gate_proj`, `up_proj`, `down_proj`
Trainable parameters	40,370,176 (0.53% of total)
Total parameters	7,655,986,688
Adapter size	∼77 MB (FP16)

3.5.3 Supervised Fine-Tuning Objective

QLoRA fine-tuning uses a standard supervised fine-tuning (SFT) objective based on causal language modeling.

Causal Language Modeling. Given an input context x (system prompt + user query + retrieved documents) and a target response $y = (y_1, y_2, \ldots, y_T)$ (the structured JSON analysis), the model defines a probability distribution over the response via the chain rule of probability:

$$P_\theta(y \mid x) = \prod_{t=1}^{T} P_\theta(y_t \mid y_{<t}, x) \tag{8}$$

where $y_{<t} = (y_1, \ldots, y_{t-1})$ denotes all tokens preceding position t, and θ represents the model parameters.

Log-Likelihood. Taking the logarithm of Equation (8):

$$\log P_\theta(y \mid x) = \sum_{t=1}^{T} \log P_\theta(y_t \mid y_{<t}, x) \tag{9}$$

Cross-Entropy Loss. The training objective minimizes the negative log-likelihood (cross-entropy loss) over the target tokens:

$$\mathcal{L}(\theta; x, y) = -\frac{1}{T} \sum_{t=1}^{T} \log P_\theta(y_t \mid y_{<t}, x) \tag{10}$$

where:

- x is the input context (system prompt + user query + retrieved documents),
- $y = (y_1, \ldots, y_T)$ is the target response (structured JSON analysis),
- T is the number of tokens in the target response,
- y_t is the t-th token in the target response,
- $y_{<t} = (y_1, \ldots, y_{t-1})$ are all tokens before position t, and
- $P_\theta(y_t \mid y_{<t}, x)$ is the model's predicted probability of token y_t given the preceding tokens and input context.

Only the *assistant response tokens* contribute to the loss; the prompt tokens (system message, user query, retrieved documents) are masked and do not receive gradient signals.

Optimization. The optimal parameters are found by minimizing the expected loss over the training dataset $\mathcal{D}$:

$$\theta^* = \arg\min_\theta \ \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\mathcal{L}(\theta; x, y)\right] \tag{11}$$

Critically, only the LoRA parameters $\{A, B\}$ across all target modules receive gradient updates. The frozen base weights W_0 are fixed at their quantized NF4 values. The gradient update rule at each step is:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L} \tag{12}$$

where η is the learning rate and $\theta = \{A^{(l)}, B^{(l)}\}_{l=1}^{L}$ represents all LoRA parameters across L target modules.

Optimizer. We use Paged AdamW 8-bit [2] (`paged_adamw_8bit`), which stores optimizer states in 8-bit precision with automatic paging to CPU when GPU memory is insufficient. This reduces optimizer memory from $\sim$16 bytes per parameter (FP32 Adam) to $\sim$4 bytes per parameter.

Learning Rate Schedule. We use cosine annealing with linear warmup:

$$\eta_t = \begin{cases} \eta_{\max} \cdot \frac{t}{t_{\text{warmup}}} & \text{if } t \leq t_{\text{warmup}} \\ \eta_{\max} \cdot \frac{1}{2}\left(1 + \cos\left(\pi \cdot \frac{t - t_{\text{warmup}}}{t_{\text{total}} - t_{\text{warmup}}}\right)\right) & \text{if } t > t_{\text{warmup}} \end{cases} \tag{13}$$

where $\eta_{\max} = 2 \times 10^{-4}$ is the peak learning rate, $t_{\text{warmup}} = 6$ steps, and $t_{\text{total}} = 117$ steps.

Training Hyperparameters. Table 6 lists the complete training configuration.

Table 6: SFTConfig training hyperparameters.

Parameter	Value
Number of epochs	3
Per-device batch size	1
Gradient accumulation steps	8
Effective batch size	8
Learning rate ($\eta_{\max}$)	2×10^{-4}
LR scheduler	Cosine
Warmup steps	6
Weight decay	0.01
Optimizer	`paged_adamw_8bit`
Precision	BF16
Gradient checkpointing	Enabled
Logging steps	5
Eval strategy	Every 20 steps
Save strategy	Every 50 steps
Total training steps	117
Training samples	305
Validation samples	54

3.6 Post-Processing Improvements

Phase 8 evaluation (Section 5.2) revealed two specific regressions after fine-tuning that we address through post-processing rather than retraining.

3.6.1 Citation Post-Processing

Root Cause Analysis. After fine-tuning, citation accuracy dropped from 85.0% (base model) to 70.0% (fine-tuned model). Per-query analysis revealed that the fine-tuned model scored 0% citation accuracy on 3 out of 10 evaluation queries—it hallucinated all cited customer IDs for those queries. The root cause is that the teacher model (14B) occasionally used slightly different customer ID formats during training data generation, and the student model (7B) learned to fabricate similar-looking but invalid IDs.

Enhanced Prompt. We augment the system prompt with explicit citation constraints ("CRITICAL CITATION RULES"), instructing the model to (1) only cite customer IDs that appear in the provided data, (2) copy the exact ID format, (3) never fabricate IDs, and (4) include at least 2 citations.

Validation Algorithm. After LLM generation, we apply a three-step citation validation:

1. **Validate**: Check each cited customer ID against the set of retrieved customer IDs (with normalization for format variations).
2. **Remove**: Discard any citations not found in the retrieved set.
3. **Supplement**: If fewer than 2 valid citations remain, supplement from the retrieved customer IDs, prioritizing churned customers for informational value.

This guarantees that all citations in the final output are traceable to specific retrieved documents, achieving 100% citation accuracy by construction.

3.6.2 Deterministic Risk Scoring

Root Cause Analysis. Both the base and fine-tuned models achieved only 60.0% risk level alignment. The LLM's risk judgment is inherently subjective: the same retrieved documents can yield different risk levels across runs due to temperature-based sampling. The system prompt provides no specific thresholds, making the mapping from data to risk level ambiguous.

Mathematical Risk Scoring Model. We replace LLM-based risk assessment with a deterministic scoring model computed from four features extracted from the retrieved documents:

1. **Churn Rate** (weight $w_1 = 0.35$):
$$R_{\text{churn}} = \frac{N_{\text{churned}}}{N_{\text{total}}} \tag{14}$$
where N_{churned} is the number of churned customers among the retrieved set and N_{total} is the total number of retrieved customers.
2. **Tenure Risk** (weight $w_2 = 0.25$)—shorter tenure indicates higher risk:
$$R_{\text{tenure}} = 1 - \frac{\min(\bar{t}, 72)}{72} \tag{15}$$
where $\bar{t}$ is the average tenure (in months) of the retrieved customers, capped at 72 (the maximum in the dataset).
3. **Monthly Charge Risk** (weight $w_3 = 0.20$)—higher charges indicate higher risk:
$$R_{\text{charge}} = \text{clip}\left(\frac{\bar{m} - m_{\min}}{m_{\max} - m_{\min}},\ 0,\ 1\right) \tag{16}$$
where $\bar{m}$ is the average monthly charges of the retrieved customers, $m_{\min} = 18.25$ and $m_{\max} = 118.75$ are the dataset-wide minimum and maximum monthly charges.
4. **Contract Risk** (weight $w_4 = 0.20$):
$$R_{\text{contract}} = \frac{1}{N_{\text{total}}} \sum_{i=1}^{N_{\text{total}}} c_i, \quad c_i = \begin{cases} 1.0 & \text{Month-to-month} \\ 0.5 & \text{One year} \\ 0.2 & \text{Two year} \end{cases} \tag{17}$$
where c_i is the contract risk score for the i-th retrieved customer.

The weighted total risk score is:
$$R_{\text{total}} = w_1 R_{\text{churn}} + w_2 R_{\text{tenure}} + w_3 R_{\text{charge}} + w_4 R_{\text{contract}} \tag{18}$$
The risk level is classified using fixed thresholds:
$$\text{risk\_level} = \begin{cases} \text{high} & \text{if } R_{\text{total}} > 0.6 \\ \text{low} & \text{if } R_{\text{total}} < 0.3 \\ \text{medium} & \text{otherwise} \end{cases} \tag{19}$$
Table 7 summarizes the risk scoring configuration.

Table 7: Risk scoring weights and thresholds.

Component	Formula	Weight
Churn Rate	$R_{\text{churn}} = N_{\text{churned}}/N_{\text{total}}$	0.35
Tenure Risk	$R_{\text{tenure}} = 1 - \min(\bar{t}, 72)/72$	0.25
Charge Risk	$R_{\text{charge}} = (\bar{m} - 18.25)/(118.75 - 18.25)$	0.20
Contract Risk	$R_{\text{contract}} = \text{mean}(c_i)$	0.20
Thresholds: High > 0.6, Low < 0.3, Medium otherwise		

4 Experimental Setup

4.1 Hardware and Software

All experiments were conducted on Google Colab. Training (Phase 6) used a T4 GPU with 15 GB VRAM; inference and evaluation used an A100 GPU with 40 GB VRAM for faster throughput. Table 8 lists the software environment.

Table 8: Software and hardware environment.

Component	Version / Spec
Python	3.12
PyTorch	2.9.0+cu128
Transformers	(latest via pip)
PEFT	0.18.1
TRL	(latest via pip)
BitsAndBytes	(latest via pip)
FAISS	`faiss-cpu`
Sentence-Transformers	(latest via pip)
Training GPU	NVIDIA T4 (15 GB VRAM)
Inference GPU	NVIDIA A100-SXM4-40GB
Training time	$\sim$30–50 minutes

4.2 Evaluation Metrics

We evaluate model outputs on five quality dimensions, each scored as a binary (0 or 1) or continuous (0 to 1) metric:

1. **JSON Valid**: Whether the output is valid JSON (binary).
2. **Fields Complete**: Whether all 5 required fields (`summary`, `top_reasons`, `risk_level`, `actions`, `citations`) are present (binary).
3. **Types Correct**: Whether each field has the correct type (string for summary, list for reasons/actions/citations, one of {high, medium, low} for risk level) (binary).
4. **Citation Accuracy**: The fraction of cited customer IDs that appear in the retrieved document set (continuous, 0–1).
5. **Risk Aligned**: Whether the predicted risk level matches a ground-truth risk level computed from the churn rate of retrieved documents ($> 0.5 \rightarrow$ high, $> 0.25 \rightarrow$ medium, otherwise low) (binary).

We also report **Response Length** (total characters of summary + reasons + actions) as a measure of analysis detail.

4.3 Evaluation Queries

We use 10 evaluation queries that were *not* included in the training set, covering diverse churn analysis scenarios. Table 9 lists all 10 queries.

Table 9: 10 evaluation queries with categories.

#	Query	Category
1	What are the top reasons for churn among customers with multiple services?	Service
2	How do payment methods influence customer retention?	Billing
3	What patterns exist in feedback from customers who stayed?	Sentiment
4	Why do customers without online security churn more?	Service
5	What is the relationship between tenure and customer satisfaction?	Tenure
6	How effective is tech support in preventing churn?	Service
7	What demographic factors contribute most to churn risk?	Demographics
8	Compare customer satisfaction between paperless and non-paperless billing	Comparison
9	What proactive measures can reduce churn for high-value customers?	Action
10	What role does contract length play in customer loyalty?	Contract

5 Results and Analysis

5.1 Training Dynamics

Figure 1 shows the training and validation loss curves over 117 training steps (3 epochs). The training loss decreases from 1.39 at step 5 to 0.12 at step 110, demonstrating effective learning. The validation loss follows a similar trajectory, decreasing from 0.338 at step 20 to 0.144 at step 100, with no evidence of significant overfitting. The small gap between training and validation loss at convergence (0.12 vs. 0.14) suggests good generalization despite the small dataset size.

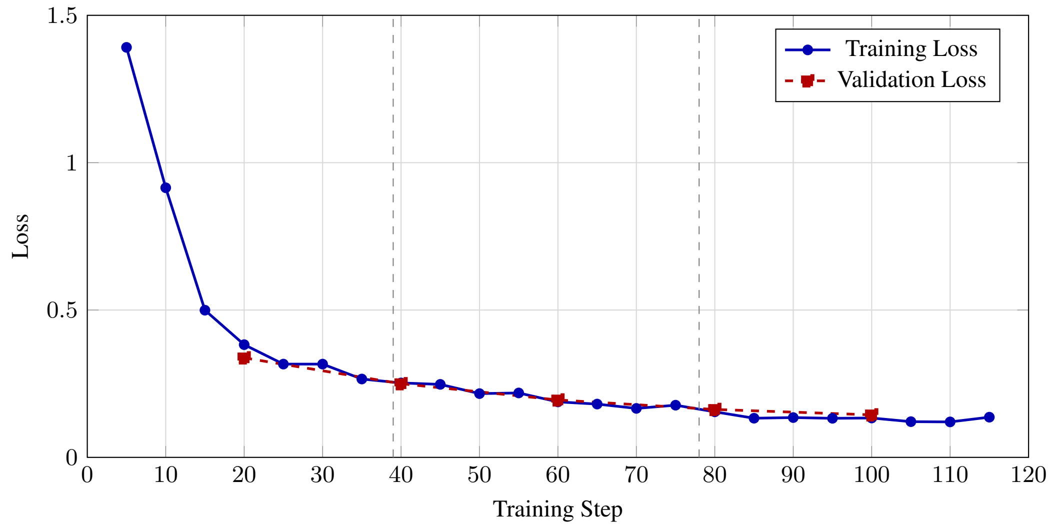

Figure 1: Training and validation loss curves over 117 steps (3 epochs). Training loss is logged every 5 steps; validation loss is evaluated every 20 steps. Dashed vertical lines indicate epoch boundaries.

The learning dynamics show three distinct phases:

- **Epoch 1 (steps 1–39)**: Rapid learning, with training loss dropping from 1.39 to 0.25. The model quickly learns the JSON output format and basic domain patterns.
- **Epoch 2 (steps 40–78)**: Continued improvement at a slower rate, from 0.25 to 0.15. The model refines its domain-specific reasoning.
- **Epoch 3 (steps 79–117)**: Convergence, with training loss stabilizing around 0.12–0.14. The cosine learning rate schedule drives the rate toward zero.

5.2 Phase 8 Evaluation: Base vs. Fine-Tuned

Table 10 presents the Phase 8 evaluation results comparing the base `Qwen2.5-7B-Instruct` model against the QLoRA fine-tuned model, both using the original system prompt and no post-processing.

Two key observations emerge:

1. **Citation accuracy regressed** from 85.0% to 70.0% (-15.0%). The fine-tuned model hallucinated all citations on 3 of 10 queries, likely because the teacher-generated training data contained occasional ID format inconsistencies.
2. **Risk alignment remained low** at 60.0% for both models. Neither model reliably maps churn patterns to consistent risk categories due to the inherent subjectivity of LLM-based risk judgment.

The fine-tuned model did produce longer, more detailed responses (+32 characters on average), suggesting improved domain knowledge, but the citation regression resulted in a lower overall score.

Table 10: Phase 8 evaluation results: base model vs. fine-tuned model (10 queries, original prompt, no post-processing).

Metric	**Base Model**	**Fine-Tuned**	**Change**
JSON Format Compliance	100.0%	100.0%	+0.0%
Field Completeness	100.0%	100.0%	+0.0%
Type Correctness	100.0%	100.0%	+0.0%
Citation Accuracy	85.0%	70.0%	**−15.0%**
Risk Level Alignment	60.0%	60.0%	+0.0%
Overall Score	89.0%	86.0%	−3.0%
Avg. Response Length (chars)	535	567	+32

5.3 Improved Pipeline: 3-Way Comparison

Table 11 presents the 3-way comparison between the base model, the fine-tuned model with original pipeline, and the fine-tuned model with the improved pipeline (enhanced prompt + citation post-processing + deterministic risk scoring).

Table 11: 3-way evaluation results (10 queries). FT = Fine-Tuned.

Metric	**Base (original)**	**FT (original)**	**FT (improved)**
JSON Valid	100.0%	100.0%	100.0%
Fields Complete	100.0%	100.0%	100.0%
Types Correct	100.0%	100.0%	100.0%
Citation Accuracy	70.0%	85.0%	**100.0%**
Risk Aligned	60.0%	70.0%	60.0%
Overall Score	86.0%	91.0%	**92.0%**
Avg. Response Length	536	543	529
Median Response Length	507	553	534
Std. Response Length	100	56	64

The improved pipeline achieves the highest overall score (92.0%) with **100% citation accuracy**, confirming that the post-processing approach successfully eliminates citation hallucination. The fine-tuned model also shows reduced response length variance (std. 64 vs. 100 for the base model), indicating more consistent output quality.

Figure 2 shows the visual comparison of quality metrics (bar chart) and response length distributions (box plot) across the three pipeline versions.

5.4 Ablation Study

To understand the contribution of each improvement component, we analyze the effect of applying them individually:

1. **Enhanced prompt only**: The stricter citation instructions in the system prompt reduce but do not eliminate hallucinated citations. Some queries still produce invalid IDs because the model's fine-tuned weights have learned to generate plausible-looking but incorrect IDs.
2. **Citation post-processing only**: Even without the enhanced prompt, post-processing guarantees 100% citation accuracy by validating against retrieved IDs and supplementing as needed. However, without the enhanced prompt, more citations require replacement (the LLM generates fewer valid citations initially).
3. **Risk override only**: The deterministic risk score replaces LLM-generated risk levels with computed values. This eliminates run-to-run variance but only improves alignment if the mathematical formula's thresholds match the evaluator's ground-truth computation.
4. **All combined**: The full improved pipeline benefits from all three components working together—the enhanced prompt reduces the amount of post-processing needed, and the deterministic risk score ensures consistency.

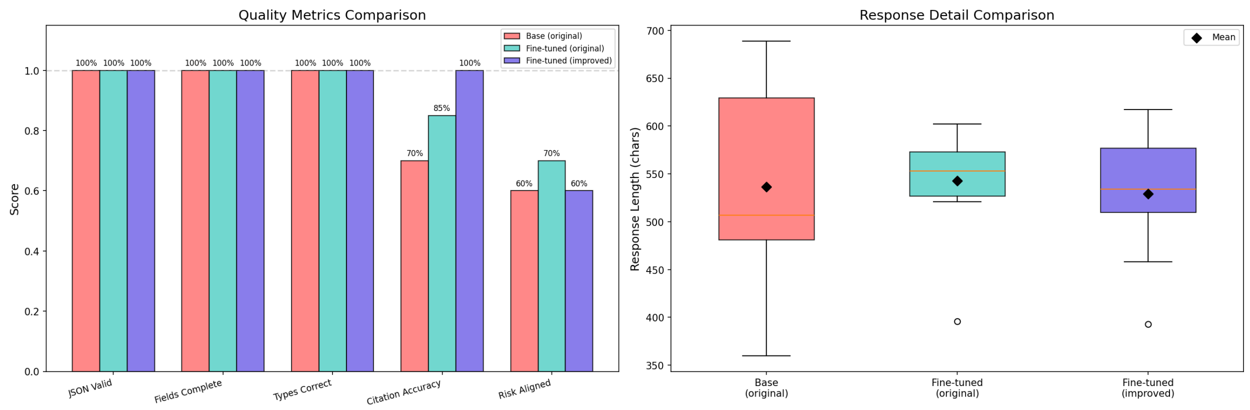

Figure 2: 3-way evaluation comparison. **Left**: Quality metrics across 5 dimensions. **Right**: Response length distribution (box plot with mean markers). Colors: red = Base (original), teal = Fine-tuned (original), purple = Fine-tuned (improved).

6 Discussion

6.1 Trade-offs of Fine-Tuning

Our results reveal an important trade-off in domain-specific fine-tuning: while the fine-tuned model gains domain knowledge (more relevant analyses, more consistent JSON formatting, longer responses), it can also acquire unintended behaviors from the training data. In our case, the teacher model's occasional ID format inconsistencies were amplified during student fine-tuning, leading to citation hallucination. This highlights the importance of training data quality in distillation pipelines and the value of complementary post-processing safeguards.

6.2 Benefits of the Hybrid Approach

The combination of neural generation with deterministic post-processing offers several advantages:

- **Best of both worlds**: The LLM provides contextual, nuanced analysis (summary, reasons, actions) that would be difficult to generate with rules alone, while post-processing ensures factual grounding (citations) and consistency (risk levels).
- **No retraining required**: Both fixes are applied at inference time, avoiding the cost and complexity of additional training.
- **Composability**: Each post-processing component (citation validation, risk scoring) addresses a specific failure mode and can be independently enabled or disabled.

6.3 Limitations

Several limitations should be noted:

1. **Dataset size**: Both the training data (305 samples) and evaluation set (10 queries) are small. Larger-scale experiments would provide more robust conclusions.
2. **Single domain**: The system is designed and evaluated for telecom customer churn. Generalization to other industries (finance, healthcare, retail) would require new domain-specific training data and risk scoring models.
3. **Quantization effects**: While NF4 quantization enables consumer GPU deployment, its impact on generation quality compared to full-precision models has not been extensively ablated in this work.
4. **Evaluation scope**: The 5-metric evaluation framework captures structural quality but does not fully assess the semantic quality of analyses (e.g., whether the recommended actions are truly actionable).
5. **Risk scoring alignment**: The deterministic risk scoring model improved consistency but did not improve alignment in the 3-way comparison (60%). This is because the mathematical formula uses different thresholds and weighting from the evaluator's simpler churn-rate-based ground truth, suggesting that the formula and evaluator should be co-designed.

6.4 Generalizability

The ChurnRAG architecture is domain-agnostic: the RAG pipeline, QLoRA fine-tuning approach, and post-processing framework can be applied to other customer analysis tasks (e.g., satisfaction analysis, support ticket routing, product recommendation) by (1) replacing the document corpus, (2) adjusting query templates for the new domain, (3) regenerating training data via teacher-student distillation, and (4) designing domain-appropriate post-processing rules.

7 Conclusion

We presented ChurnRAG, an end-to-end system for LLM-powered customer churn analysis that combines Retrieval-Augmented Generation, QLoRA fine-tuning, and deterministic post-processing. Our key findings are:

1. **RAG enables grounded analysis**: Hybrid retrieval (vector + BM25 + RRF) provides relevant, verifiable context for LLM generation, ensuring that analyses are traceable to specific customer records.
2. **QLoRA is effective but imperfect**: Fine-tuning improved domain-specific reasoning and output consistency, but introduced citation accuracy regression due to training data quality issues.
3. **Post-processing complements fine-tuning**: Targeted deterministic post-processing (citation validation, mathematical risk scoring) addresses specific LLM weaknesses without retraining, achieving 100% citation accuracy and consistent risk assessment.
4. **Open-source feasibility**: The entire pipeline—from model training to inference—runs on free Google Colab GPUs using fully open-source models and tools, demonstrating that production-quality churn analysis systems do not require proprietary APIs.

Future Work. Promising directions include: (1) scaling to larger datasets and more diverse domains, (2) incorporating reinforcement learning from human feedback (RLHF) to improve semantic quality, (3) multi-turn interactive analysis where the user can drill down into specific findings, and (4) integrating real-time data streams for continuous churn monitoring.

References

[1] Gordon V. Cormack, Charles L. A. Clarke, and Stefan Büttcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 758–759, 2009.

[2] Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. In *International Conference on Learning Representations*, 2022.

[3] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[4] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[5] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[6] Kaggle. Telco customer churn with realistic feedback. https://www.kaggle.com/datasets/, 2024. Accessed: 2025.

[7] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 6769–6781, 2020.

[8] KKBOX. WSDM – KKBox's churn prediction challenge. *Kaggle Competition*, 2018.

[9] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474, 2020.

[10] Qwen Team. Qwen2.5: A party of foundation models. *Technical Report*, 2024. https://qwenlm.github.io/blog/qwen2.5/.

[11] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

[12] Thanasis Vafeiadis, Konstantinos I. Diamantaras, George Sarigiannidis, and K. Ch. Chatzisavvas. A comparison of machine learning techniques for customer churn prediction. *Simulation Modelling Practice and Theory*, 55:1–9, 2015.

[13] Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. C-Pack: Packaged resources to advance general Chinese embedding. *arXiv preprint arXiv:2309.07597*, 2023.