Automated Financial Report Reasoning with GRPO

Training Verifiable AI Analysts at QuantaLedger Analytics

Section 1: Industry Context and Business Problem

Industry Overview

The financial services industry generates approximately 2.5 million earnings reports, regulatory filings, and analyst notes annually across publicly traded companies in the United States alone. Each document contains structured numerical data (revenue figures, margins, ratios) alongside narrative prose that contextualizes performance.

Institutional investors, hedge funds, and compliance teams must extract accurate quantitative answers from these documents under time pressure. A single misread figure -- confusing quarterly revenue with annual revenue, or misidentifying the comparison period -- can lead to erroneous trading decisions worth millions.

Company Profile

QuantaLedger Analytics is a Series B fintech startup (\$42M raised) headquartered in Chicago. They provide an AI-powered financial document analysis platform used by 85 mid-tier hedge funds and 12 compliance departments at regional banks. Their core product is an automated Q&A system: analysts type natural-language questions about financial documents, and the system returns precise numerical answers with source citations.

Business Challenge

QuantaLedger's current system uses a fine-tuned LLM that was trained with supervised learning on 50,000 human-annotated question-answer pairs from financial documents. The system performs well on straightforward lookup queries ("What was Q3 revenue?") but struggles with multi-step reasoning:

- **Calculation questions**: "What was the year-over-year revenue growth rate?" (requires finding two numbers and computing a percentage)
- **Comparison questions**: "Did operating margin improve or decline compared to the previous quarter?" (requires identifying margins from two periods)
- **Conditional questions**: "If the company maintains its current growth rate, what would projected Q4 revenue be?" (requires reasoning over trends)

Internal benchmarks show: - Simple lookup accuracy: 91% - Single-step calculation accuracy: 67% - Multi-step reasoning accuracy: 43%

Clients are churning because the 43% accuracy on reasoning queries makes the system unreliable for the exact use cases that justify the \$8,000/month subscription. QuantaLedger needs to improve multi-step reasoning accuracy to at least 75% without increasing inference latency beyond 2 seconds.

Constraints

- **Model size**: The production model is 7B parameters, running on 2x A100 GPUs. Memory is tight.
- **Verifiability**: Financial answers are numerically verifiable -- the system can check if the computed answer matches the ground truth extracted from documents.
- **No reward model budget**: Training a separate reward model would require an additional \$80K in annotation costs and 2 months of development.
- **Latency**: Inference must complete within 2 seconds per query.

Why GRPO

GRPO is the ideal solution for this problem because:

1. **Verifiable rewards**: Financial calculations have ground-truth answers. A reward model is unnecessary -- just check if the number is correct.
2. **No critic needed**: The 7B model already strains GPU memory. Adding a 7B critic would require 4x A100s instead of 2x.
3. **Multi-step reasoning**: DeepSeek-R1 showed that GRPO can induce chain-of-thought reasoning from binary correctness signals alone.
4. **Simplicity**: The engineering team is small (6 ML engineers). GRPO is simpler to implement than PPO.

Section 2: Technical Problem Formulation

Problem Type

This is a **reinforcement learning problem** where we optimize a language model's policy to maximize the probability of generating correct numerical answers to financial reasoning questions. The problem is specifically suited to GRPO because:

- Rewards are binary and verifiable (correct answer or not)
- The task requires multi-step reasoning (not just pattern matching)
- Memory constraints prohibit a separate critic network

- Group sampling enables learning from contrasts between correct and incorrect reasoning paths

Input/Output Specifications

Input: A financial document context C (up to 2048 tokens) and a natural-language question q.

Output: A reasoning chain followed by a numerical answer, formatted as:

```
<think>
Step 1: Identify Q3 2024 revenue: $2.3B
Step 2: Identify Q3 2023 revenue: $1.9B
Step 3: Compute growth rate: (2.3 - 1.9) / 1.9 = 0.4 / 1.9 = 0.2105
Step 4: Convert to percentage: 21.05%
</think>
Answer: 21.1%
```

Mathematical Foundation

The GRPO objective for this problem:

$$\mathcal{J}(\theta) = \mathbb{E}_{(C,q)\sim\mathcal{D}}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left(\min(r_{i,t}\hat{A}_i, \text{clip}(r_{i,t}, 1-\epsilon, 1+\epsilon)\hat{A}_i\right.\right.$$

where: - (C, q) is a context-question pair from the financial dataset $\mathcal{D}$ - $G = 16$ completions are sampled per question - $r_{i,t} = \pi_\theta(o_{i,t}|C, q, o_{i,<t})/\pi_{\theta_{\text{old}}}(o_{i,t}|C, q, o_{i,<t})$ is the per-token importance ratio - $\hat{A}_i = (r_i - \mu)/\sigma$ is the group-relative advantage with binary reward $r_i \in \{0, 1\}$

Reward Function

The reward function combines correctness and format:

$$r_i = r_{\text{correct}}(o_i, a^*) + \alpha \cdot r_{\text{format}}(o_i)$$

Correctness reward (r_{correct}):

$$r_{\text{correct}} = \begin{cases} 1 & \text{if } |\text{extracted\_answer}(o_i) - a^*| < \delta \\ 0 & \text{otherwise} \end{cases}$$

where a^* is the ground-truth answer and δ is a tolerance (e.g., 0.5% for percentages, \$0.01B for dollar amounts).

Format reward (r_{format}):

$$r_{\text{format}} = \begin{cases} 0.1 & \text{if } o_i \text{ contains } \texttt{<think>...</think>} \text{ and } \texttt{Answer:} \\ 0 & \text{otherwise} \end{cases}$$

This encourages the model to show its reasoning steps.

Let us verify with numbers. Suppose $G = 4$ with rewards $\{1.1, 0.0, 1.1, 0.1\}$ (two correct with format, one wrong with format, one wrong without):

- $\mu = (1.1 + 0.0 + 1.1 + 0.1)/4 = 0.575$
- $\sigma = \sqrt{((1.1 - 0.575)^2 + (0 - 0.575)^2 + (1.1 - 0.575)^2 + (0.1 - 0.575)^2)/4} = \sqrt{0.276 + 0.331 + 0.276 + 0.226)/4} \approx 0.527$
- $\hat{A}_1 = (1.1 - 0.575)/0.527 = +0.996$ (correct + formatted: reinforced)
- $\hat{A}_2 = (0.0 - 0.575)/0.527 = -1.091$ (wrong + no format: penalized)

Loss Function

The total training loss has three terms:

$$\mathcal{L} = -\mathcal{L}_{\text{policy}} + \beta \mathcal{L}_{\text{KL}}$$

Term 1 -- Clipped policy objective ($\mathcal{L}_{\text{policy}}$): Maximizes the probability of correct completions while clipping to prevent instability. The clipping threshold $\epsilon = 0.2$ ensures updates are conservative.

Justification: Without clipping, a single high-advantage response could cause a catastrophic policy collapse. In financial applications, stability is critical.

Term 2 -- KL divergence penalty ($\mathcal{L}_{\text{KL}}$): Prevents the model from forgetting its pre-trained knowledge. $\beta = 0.04$ balances exploration with knowledge preservation.

Justification: Financial language understanding was learned during pre-training and SFT. GRPO should add reasoning ability without erasing the model's ability to parse financial documents.

Evaluation Metrics

Metric	Description	Target
Exact Match Accuracy	Fraction of answers within tolerance	> 75%
Reasoning Step Accuracy	Fraction of correct intermediate steps	> 80%
Format Compliance	Fraction using `<think>` tags	> 90%
Latency (p95)	95th percentile inference time	< 2s
KL Divergence	Policy drift from reference	< 5.0

Baseline

The SFT-only model achieves: - Exact match on multi-step: 43% - Format compliance: 0% (no chain-of-thought training) - Latency: 1.2s

---

Section 3: Implementation Notebook Structure

3.1 Data Pipeline

```
class FinancialReasoningDataset:
    """
    Dataset of (context, question, answer) triples from financial documents.
    Answers are numerically verifiable.
    """

    def __init__(self, data_path: str, max_context_len: int = 2048):
        # TODO: Load and preprocess financial QA dataset
        # Hint: Each sample has 'context', 'question', 'answer', 'answer_type'
        # answer_type in ['percentage', 'dollar', 'ratio', 'integer']
        pass

    def __getitem__(self, idx):
        # TODO: Return (context, question, ground_truth_answer, answer_type)
        pass

    def verify_answer(self, predicted: str, ground_truth: str, answer_type: str) -> float:
        """
        TODO: Implement answer verification with type-aware tolerance.

        Hints:
        - percentage: tolerance = 0.5 percentage points
        - dollar: tolerance = 0.01B
        - ratio: tolerance = 0.01
        - integer: exact match
        """
        pass
```

3.2 Exploratory Data Analysis

```
def analyze_dataset(dataset):
    """
    TODO: Analyze the financial reasoning dataset.

    Compute and visualize:
    1. Distribution of answer types (percentage, dollar, ratio, integer)
    2. Distribution of reasoning steps required (1-step, 2-step, 3+ step)
    3. Context length distribution
    4. Question complexity categorization

    Hint: Use matplotlib bar charts and histograms.
    """
    pass
```

3.3 Baseline Evaluation

```
def evaluate_baseline(model, tokenizer, dataset, n_samples=200):
    """
    TODO: Evaluate the SFT-only model on the financial reasoning dataset.

    For each sample:
    1. Format the prompt: context + question
    2. Generate a completion (greedy decoding)
    3. Extract the numerical answer
    4. Compare with ground truth using verify_answer()

    Track: accuracy by question type, average latency, error patterns.
    """
    pass
```

3.4 GRPO Model Architecture

```
class GRPOFinancialModel:
    """
    TODO: Wrap the base model for GRPO training.

    Components:
    - policy_model: The 7B model being trained
    - ref_model: Frozen copy of the initial model
    - tokenizer: For encoding/decoding

    Methods:
    - generate_group(prompt, G): Generate G completions
    - get_log_probs(input_ids): Compute per-token log probabilities
    """

    def __init__(self, model_path: str):
        # TODO: Load model and create reference copy
        # Hint: Use AutoModelForCausalLM.from_pretrained()
        # Hint: ref_model should have requires_grad_(False)
        pass

    def generate_group(self, prompt_ids, G=16, max_new_tokens=256):
        # TODO: Generate G diverse completions using sampling
        # Hint: Use temperature=1.0 for diversity
        pass

    def get_log_probs(self, input_ids, target_ids):
        # TODO: Forward pass + gather log probs of target tokens
        pass
```

3.5 GRPO Training Loop

```
def grpo_training_step(model, dataset, optimizer, G=16, epsilon=0.2, beta=0.04):
    """
    TODO: Implement one GRPO training step for financial reasoning.

    Steps:
    1. Sample a (context, question, answer) from dataset
    2. Generate G completions from current policy
    3. Compute rewards: correctness + format
    4. Compute group-relative advantages
    5. Compute GRPO loss (clipped surrogate + KL)
    6. Backpropagate and update

    Return: loss, mean_reward, accuracy, kl_divergence
    """
    pass
```

3.6 Training Evaluation

```
def evaluate_during_training(model, dataset, n_samples=50):
    """
    TODO: Evaluate the model during training.

    Compute:
    - Multi-step reasoning accuracy (the key metric)
    - Single-step accuracy (should not degrade)
    - Format compliance rate
    - KL divergence from reference

    Hint: Use greedy decoding (temperature=0) for evaluation.
    """
    pass
```

3.7 Error Analysis

```
def error_analysis(model, dataset, n_samples=100):
    """
    TODO: Analyze the model's errors to identify patterns.

    Categories:
    1. Extraction error: wrong number pulled from context
    2. Calculation error: right numbers, wrong arithmetic
    3. Reasoning error: wrong approach/formula
    4. Format error: answer not properly structured
    5. Hallucination: uses numbers not in the context

    Visualize: confusion matrix of error types by question complexity.
    """
    pass
```

3.8 Deployment Preparation

```
def prepare_for_deployment(model, tokenizer, max_latency_ms=2000):
    """
    TODO: Prepare the GRPO-trained model for production.

    Steps:
    1. Merge LoRA weights (if using LoRA)
    2. Quantize to INT8 for inference
    3. Benchmark latency on sample queries
    4. Verify accuracy on held-out test set
    5. Export model artifacts

    Hint: Use torch.quantization or bitsandbytes for quantization.
    """
    pass
```

3.9 Ethics and Limitations

```
def ethics_assessment():
    """
    TODO: Document ethical considerations.

    Address:
    1. Hallucination risk: model might generate plausible but wrong numbers
    2. Overconfidence: model does not express uncertainty
    3. Adversarial inputs: manipulated financial documents
    4. Bias: performance across different industries/sectors
    5. Regulatory: compliance with financial advisory regulations
    6. Human oversight: when should the system defer to a human analyst?

    Output: structured risk assessment table.
    """
    pass
```

Section 4: Production and System Design Extension

System Architecture

The production system consists of three layers:

1. **Document Processing Layer**: Ingests PDF/HTML financial documents, extracts text and tables, and creates structured context chunks.

2. **Reasoning Layer**: The GRPO-trained model receives (context, question) pairs and generates reasoning chains with numerical answers.

3. **Verification Layer**: Post-processing that validates format compliance, checks numerical plausibility (e.g., revenue should be positive), and flags low-confidence answers.

API Design

```
POST /api/v2/analyze
{
    "document_id": "SEC_10K_AAPL_2024Q3",
    "question": "What was the year-over-year revenue growth rate?",
    "max_tokens": 256,
    "require_reasoning": true
}

Response:
{
    "answer": "21.1%",
    "reasoning": "Step 1: Q3 2024 revenue: $2.3B...",
    "confidence": 0.87,
    "source_citations": ["page 4, line 12", "page 4, line 28"],
    "latency_ms": 1450
}
```

Model Serving

- **Infrastructure**: 2x A100-80GB GPUs behind a load balancer
- **Framework**: vLLM for efficient batch inference with PagedAttention
- **Quantization**: INT8 post-training quantization (reduces memory by ~50%)
- **Batching**: Dynamic batching with max batch size 8, max wait time 100ms

Monitoring

Metric	Threshold	Action
Accuracy (rolling 1hr)	< 70%	Alert + investigate
Latency p95	> 2000ms	Scale up instances
KL divergence	> 5.0	Model drift -- retrain
Format compliance	< 85%	Check prompt template
Error rate (5xx)	> 1%	Page on-call

Drift Detection

Monitor for two types of drift:

1. **Data drift**: New financial document formats, terminology changes, new reporting standards (e.g., IFRS updates). Detect by tracking vocabulary overlap between training and production queries.

2. **Performance drift**: Accuracy degradation over time as the financial landscape evolves. Detect by sampling 5% of production queries for human verification.

A/B Testing

- **Control**: SFT-only model (current production)
- **Treatment**: GRPO-trained model
- **Allocation**: 20% treatment, 80% control (conservative rollout)
- **Primary metric**: Multi-step reasoning accuracy on human-verified samples
- **Duration**: 2 weeks minimum, 1000+ verified samples
- **Guardrails**: Automatic rollback if accuracy drops below SFT baseline on any question type

CI/CD Pipeline

1. **Data pipeline**: Daily ingestion of new financial documents
2. **Evaluation pipeline**: Weekly evaluation on held-out test set
3. **Retraining trigger**: When accuracy on new documents drops below 70%
4. **GRPO retraining**: 4-hour training run on 8x A100 cluster
5. **Staged rollout**: Canary (5%) -> Shadow (50%) -> Production (100%)
6. **Rollback**: Automated if accuracy drops > 3% from baseline

Cost Analysis

Component	Monthly Cost
Training (monthly retrain)	\$2,400
Inference (2x A100)	\$6,200
Data pipeline	\$800
Monitoring	\$300
Total	**\$9,700**

Compared to PPO-based training: - PPO would require 4x A100 (policy + critic): \$12,400/month inference - PPO training requires critic: \$4,800/month training - **GRPO saves ~\$7,100/month** (42% cost reduction)

References

1. Shao, Z. et al. "DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models." arXiv:2402.03300 (2024).
2. DeepSeek-AI. "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning." arXiv:2501.12948 (2025).

3. Schulman, J. et al. "Proximal Policy Optimization Algorithms." arXiv:1707.06347 (2017).