

FinDialogLens: Event Extraction over Multi-Party Dialogue for Missed-Trade Identification in Financial Chatrooms

Chin-Lun Fu Hong Ni Behrouz Madahian

Machine Learning Center of Excellence, JPMorgan Chase & Co.

{chin-lun.fu,hong.ni,behrouz.madahian}@jpmchase.com

Abstract

Multi-party financial chatrooms are vital for sales-and-trading professionals, but their complexity makes manual recovery of missed trades infeasible: each *Request for Quote* (RFQ) is an event whose final price and trade outcome appear many messages after the *RFQ-trigger message* (the inquiry message), interleaved with concurrent RFQs from other participants. We cast this as **event extraction (EE) over multi-party dialogue** and present **FinDialogLens**, a hybrid LLM pipeline in which compact fine-tuned classifiers act as inference-time scaffolds: they detect RFQ-triggers and price/trade outcome metadata, an RFQ-Level Module segments per-event RFQ windows, and a Trade Engine fills argument roles. With GPT-4o, FinDialogLens reaches 92.1% and 94.3% accuracy on final price and trade outcome, respectively, outperforming full-chatroom CoT prompting methods; fine-tuned open-source LLMs with as few as 3B parameters achieve comparable performance with modest in-domain data. To make the LLM-based solution practical at scale, a difficulty-aware router balances cost and accuracy by allocating RFQs between a low-cost rule-based engine and the higher-performing LLM-powered Trade Engine, cutting LLM calls by 85% on final price while recovering half of the accuracy gap to FinDialogLens (GPT-4o), saving over \$300/day at our 70,000-RFQ/day scale.

1 Introduction

Financial institutions communicate with their clients at scale over multi-party chatrooms, where a single day routinely carries hundreds of messages and tens of distinct trade negotiations posted concurrently. A typical interaction begins when a client reaches out with a *Request for Quote* (RFQ), e.g., “can I get a price quote on 5Y XYZ?”; a sales representative responds with a quote (e.g., “75 / 76”), and the client either agrees (e.g., “mine”) or walks away (e.g., “away, thanks”). The inquiry, the final agreed

price, and the trade outcome together form the structured record of the business event. In practice, RFQ negotiations are interleaved, asynchronous, and noisy: quotes, requotes, and trade outcomes may be separated by minutes, mixed with parallel client negotiations, and obscured by irrelevant market updates or chitchat. Consequently, identifying the trigger, quotes, and final outcome for a given RFQ requires linking distant, non-adjacent messages across substantial irrelevant context.

Executed trades are logged automatically by systems, but *missed* trades, namely inquiries that did not result in execution, are not, leaving sales desks unable to measure quote competitiveness or prioritize follow-up. At production volumes of tens of thousands of RFQs per day, manual recovery of missed-trade records from chat is infeasible.

We cast this problem as **event extraction over multi-party dialogue**, an application of EE to a setting where each RFQ is an event whose *trigger* is the inquiry message, and whose argument roles, the final price and the trade outcome, are filled by content from downstream messages that often appear many minutes later and are interleaved with unrelated concurrent RFQs from other participants. This combination of long-range argument scattering and multi-party interleaving is not the setting that existing EE recipes target: sentence-level methods (Doddington et al., 2004; Li et al., 2013; Chen et al., 2015) are bounded to a single sentence, while document-level methods (Ebner et al., 2020; Li et al., 2021; Zheng et al., 2019) assume coherent single-author text. Table 1 summarizes why other prior EE systems are not directly comparable.

Our framework, **FinDialogLens** (Figure 1), is a hybrid LLM pipeline with three modules. The *Message-Level Module* runs compact RoBERTa-base (Liu et al., 2019) classifiers fine-tuned on in-domain data to produce metadata for RFQ-trigger messages, price mentions, and trade outcome cues; these predictions act as *inference-time scaffolds* for

the downstream LLM, supplying domain-specific signals that general-purpose LLMs lack on raw chat (Yu et al., 2023). The *RFQ-Level Module* segments the chatroom into per-event RFQ windows, one for each detected RFQ-trigger message, using a temporal heuristic. The *Trade Engine* performs LLM-based argument role filling over each RFQ window using GPT-4o (OpenAI et al., 2024) or a fine-tuned open-source LLM. To make the LLM-based solution practical at production scale, a *difficulty-aware router* predicts per-event difficulty and assigns easier RFQ windows¹ to a low-cost rule-based weak engine while escalating harder windows to the LLM-powered Trade Engine. Our contributions are as follows.

- We cast missed-trade identification as **event extraction (EE) over multi-party dialogue**, a setting in which arguments are unanchored, spread across non-adjacent turns, and interleaved with concurrent RFQ events. We show that direct off-the-shelf LLM prompting over full chatrooms performs poorly on this task.
- We show that **per-event RFQ windowing combined with compact specialized classifiers as inference-time scaffolds substantially improves GPT-4o over full-chatroom CoT prompting** for multi-party dialogue EE. The resulting pipeline, **FinDialogLens**, uses these classifiers as inference-time scaffolds for LLM extraction and achieves 92.1% and 94.3% accuracy for final price and trade outcome extraction.
- We propose a **difficulty-aware router** supervised by the per-event correctness of a rule-based weak engine. For final price, routing around 15% of RFQs to the LLM-powered Trade Engine (an 85% reduction in LLM calls) recovers half of the accuracy gap between the weak engine and FinDialogLens (GPT-4o), saving over \$300/day at our 70,000-RFQ/day scale.

2 Related Work

Event extraction and financial NLP. Event extraction (EE) has progressed from sentence-level formulations on ACE (Doddington et al., 2004) to document-level argument linking (Ebner et al.,

¹Here, “easier” denotes RFQ windows that follow typical conversational patterns and can be addressed reliably using heuristic rules.

Method family	Multi-party	Cross-turn	Trigger-anchored	Event record
Sentence EE	×	×	✓	✓
Doc-level EE	×	✓	×	✓
Dialogue disent.	✓	✓	–	×
Ours	✓	✓	✓	✓

Table 1: Structural axes along which existing EE method families differ from our application setting.

2020; Li et al., 2021; Zheng et al., 2019) and generative or instruction-tuned methods (Lu et al., 2021; Hsu et al., 2022; Wang et al., 2023; Sainz et al., 2024). Financial NLP (Li et al., 2024b; Nie et al., 2024; Chen et al., 2024) applies structured information extraction (IE) primarily to news and filings (e.g. Doc2EDAG (Zheng et al., 2019), FinRED (Sharma et al., 2023)); conversational financial data receives little attention. Dialogue disentanglement (Kummerfeld et al., 2019; Li et al., 2024a) handles multi-party input but targets thread separation rather than event extraction. Our setting requires four structural properties simultaneously: multi-party input, arguments that span multiple turns, explicit per-event triggers, and structured event records as output.

3 Methodology

3.1 Problem Setting

A chatroom $C = (m_1, \dots, m_N)$ is a time-ordered sequence of messages, each tagged with an author identifier. We model each RFQ as a typed event anchored by a single *RFQ-trigger message* and associated with two argument roles: *final price* and *trade outcome*. Throughout, an RFQ-trigger message refers to the message that initiates an RFQ. The goal is to recover the set of K RFQ events present in C ,

$$E = \{e_j\}_{j=1}^K, \quad e_j = (m_j^{\text{trig}}, p_j, o_j), \quad (1)$$

where $m_j^{\text{trig}} \in C$ is the RFQ-trigger message that anchors the event, p_j is the final price, represented as a string lifted from the dialogue or NONE if no price applies, and $o_j \in \{\text{Hit}, \text{Miss}, \text{Unknown}\}$ is the trade outcome.² Evidence for the two argument roles may appear well after m_j^{trig} and can be inter-

²HIT: the client agreed to trade; MISS: the client explicitly declined (e.g. “away”); UNKNOWN: no client response was observed within the context window. UNKNOWN is operationally distinct from MISS because sales follow-up decisions differ in the two cases.

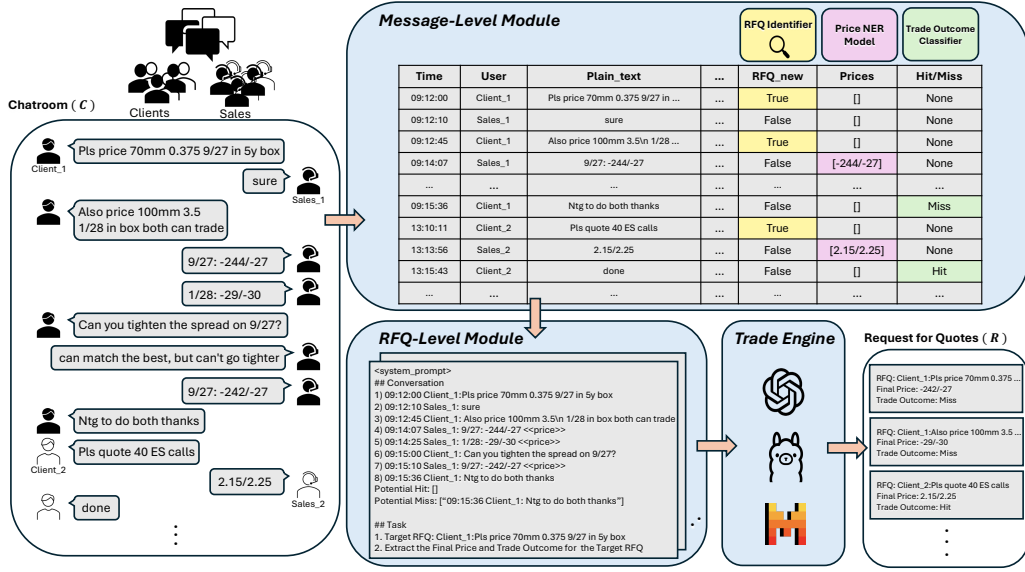


Figure 1: Overview of the FinDialogLens pipeline for event extraction over multi-party dialogue. The Message-Level Module produces message-level metadata through RFQ-trigger detection, trade outcome classification, and price NER. The RFQ-Level Module uses these predictions to segment per-event RFQ windows from the chatroom. The Trade Engine performs LLM-based argument role filling over each RFQ window.

leaved with messages from other concurrent RFQ events.

3.2 Message-Level Module

The Message-Level Module produces message-level metadata that serves as inference-time scaffolding for RFQ windowing and argument role filling. Inspired by recent work (Yu et al., 2023) showing that fine-tuning compact models like RoBERTa with sufficient data can outperform larger LLM-based models in specialized tasks, we fine-tune three RoBERTa-base models on in-domain data. The *RFQ Identifier* is a binary trigger detector that predicts whether each $m_i \in C$ initiates an RFQ event. The *Trade Outcome Classifier* assigns each message a label in {Hit, Miss, None}, highlighting messages that may express trade outcomes. A *Price NER Model* extracts price entities from each message. The resulting metadata annotations are attached to each m_i and consumed by the RFQ-Level Module and the Trade Engine.³

3.3 RFQ-Level Module

A typical daily chatroom C may contain hundreds of messages and numerous RFQs, with multiple clients and sales representatives negotiating prices for different products while also exchanging unrelated market updates and informal chatter. Processing the full chatroom context is therefore both inef-

ficient and potentially unreliable, as long-context LLMs may underutilize relevant evidence when it is buried within lengthy inputs, a phenomenon commonly described as the “lost-in-the-middle” effect (Liu et al., 2024). To address this, the RFQ-Level Module constructs an *RFQ window* for each detected RFQ-trigger message. Each RFQ window is a temporally bounded, per-event context anchored at the trigger and passed to the Trade Engine for argument role filling. We denote the window for trigger m_j^{trig} as w_j .

For each trigger m_j^{trig} flagged by the RFQ Identifier, we collect the trigger message and its subsequent messages in C until either a 20-minute time gap or a 25-message cap is reached.⁴ The message-level metadata is interleaved with the raw text inside w_j using special tokens (Figure 7). The module outputs the set of RFQ windows $W = \{w_j\}$, one for each trigger detected by the RFQ Identifier.

3.4 Trade Engine

Given a per-event RFQ window $w_j \in W$ anchored at trigger message m_j^{trig} , the Trade Engine predicts the remaining argument roles: the final price p_j and trade outcome o_j . The full event record is then assembled as $e_j = (m_j^{\text{trig}}, p_j, o_j)$.

The Trade Engine is LLM-powered. It prompts

⁴Thresholds were chosen on the evaluation split; the resulting window covers the gold final price and trade outcome for over 96% of RFQ events.

³Training details are reported in Appendix B.1.

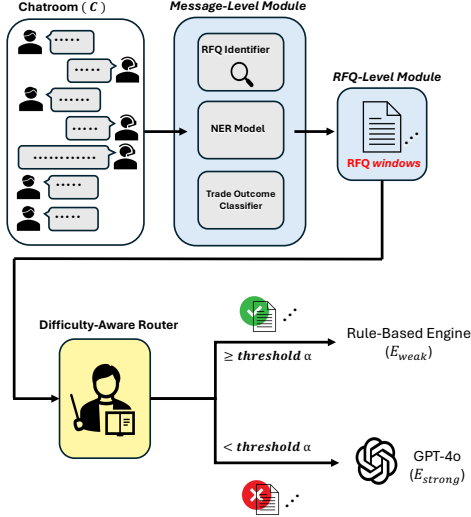


Figure 2: Difficulty-Aware Router. For each RFQ event, the router assigns easier cases to rule-based engine and escalates harder cases to the LLM-powered engine.

an LLM with the metadata-augmented RFQ window w_j under a unified template (Figure 9), then parses the structured response into (p_j, o_j) . The LLM is instructed to identify the final price when applicable and determine the trade outcome from the dialogue evidence. We instantiate the Trade Engine with two LLM families:

1. **Proprietary:** GPT-4o.
2. **Open-source:** Mistral-7B-Instruct (Jiang et al., 2023), Llama-3-8B-Instruct (Grattafiori et al., 2024), Phi-3-mini-4k-instruct (Abdin et al., 2024), and Flan-T5-XL (Chung et al., 2022).

Throughout, “FinDialogLens (M)” denotes the full pipeline using model M as the LLM-powered Trade Engine.

3.5 Difficulty-Aware Router

Although the LLM-powered Trade Engine achieves strong performance, using it for every RFQ window incurs avoidable cost and latency. Some windows exhibit empirical structural patterns, such as nearby evidence, predictable reply behavior, or explicit product and outcome references, that a rule-based weak engine E_{weak} can exploit reliably. As shown in Figure 2, we therefore introduce a *difficulty-aware router* that sends lower-ambiguity RFQ windows to E_{weak} and reserves the LLM-powered strong engine E_{strong} for more ambiguous cases.

We train the router to estimate the reliability of the weak engine, $P_{\theta}(\text{Correct} \mid w)$, using labels obtained by running the weak engine on labeled RFQ windows. At inference time, a threshold α controls the cost-accuracy trade-off:

$$R^{\alpha}(w) = \begin{cases} E_{\text{weak}}, & P_{\theta}(\text{Correct} \mid w) \geq \alpha, \\ E_{\text{strong}}, & \text{otherwise.} \end{cases} \quad (2)$$

We instantiate P_{θ} with RoBERTa-base, Mistral-7B-Instruct, and Flan-T5-XL.

4 Experiments

4.1 Experiment Setup

4.1.1 Data Overview

We annotate 47,362 messages from two years of financial conversations (2023–2024), covering 8,343 RFQs in total. The data is partitioned into three non-overlapping splits: 37,245 messages / 7,082 RFQs for training, 1,147 messages / 295 RFQs for evaluation, and 8,970 messages / 966 RFQs for testing. The splits are time-based rather than uniformly sampled: we hold out a contiguous June–July 2024 window, the most recent period in our annotated data, as the test set. The training and evaluation sets are drawn from the remaining 2023–2024 data, with no RFQ or message overlap across splits.

This setup mimics the production setting, where models are trained on historical labeled data and evaluated on new incoming conversations. The training split is used to train the Message-Level Module, Trade Engine, and difficulty-aware router, while the evaluation split is used for prompt and hyperparameter selection. All reported results are on the test split. To protect sensitive financial communications, strict privacy protocols are enforced to prevent data leakage or unauthorized access.⁵

4.1.2 Baselines

We compare FinDialogLens against both non-LLM and LLM-based baselines. As a non-LLM baseline, we implement a Rule-based Engine that uses the same message-level metadata produced by the Message-Level Module and links predicted prices and trade outcomes to RFQ events through predefined heuristics based on conversation characteristics. As LLM-based baselines, we evaluate end-to-end Chain-of-Thought

⁵Further details on data privacy, annotation guidelines, and training details are provided in Appendix A and Appendix B.

Method	w/ Metadata	Input Context		Trade Engine	Final Price	Trade Outcome	
		Whole Day	RFQ Windows		Acc	Acc	F1
Rule-based Engine	✓	✓		Rule-based	77.1	90.3	90.7
CoT Prompting		✓		GPT-4o (Zero-shot)	60.9	65.5	77.1
		✓		GPT-4o (Few-shot)	64.4	70.5	80.5
	✓	✓		GPT-4o (Zero-shot)	73.9	85.9	91.5
	✓	✓		GPT-4o (Few-shot)	79.2	87.0	92.3
FinDialogLens	✓		✓	GPT-4o	92.1	94.3	94.5
	✓		✓	Mistral-7B-Instruct (Fine-tuned)	89.0	92.7	94.6
	✓		✓	Llama-3-8B-Instruct (Fine-tuned)	87.1	90.3	92.8
	✓		✓	Phi-3-mini-4k-instruct (Fine-tuned)	87.8	91.2	92.9
	✓		✓	Flan-T5-XL (Fine-tuned)	91.7	92.8	94.4

Table 2: Comparison of final price and trade outcome extraction performance across the Rule-based Engine, full-chatroom CoT prompting baselines, and FinDialogLens. “w/ Metadata” indicates whether the input context is augmented with predicted RFQ-trigger, price, and trade-outcome tags from the Message-Level Module. “Input Context” specifies whether the model consumes the full daily chatroom or per-event RFQ windows produced by the RFQ-Level Module.

(CoT) prompting (Wei et al., 2022) with GPT-4o (gpt-4o-2024-05-13). We run CoT prompting on the full daily chatroom under four configurations: {zero-shot, few-shot (3 examples)} $\times$ {with metadata, without metadata}. In the metadata setting, the full chatroom is augmented with the Message-Level Module’s predicted RFQ-trigger, price, and trade-outcome tags using the same special tokens as FinDialogLens (Figure 7). The CoT prompt templates are shown in Figure 8.

4.1.3 Evaluation Protocol and Metrics

We evaluate FinDialogLens with a trigger-anchored event extraction protocol. The RFQ-trigger message defines the identity of an RFQ event, while final price and trade outcome are arguments whose correctness is defined relative to that anchor. At inference time, FinDialogLens runs end-to-end from raw chatroom messages: it detects RFQ triggers and metadata, constructs one RFQ window for each detected trigger, and predicts the final-price and trade-outcome arguments. Gold triggers are not provided as input.

For scoring, predicted RFQ events are aligned to reference events by their trigger message, and argument accuracy is computed on the aligned events. RFQ-trigger detection is evaluated separately through the RFQ Identifier in the Message-Level Module, with precision, recall, and F1 reported in Table 3; this separates event-anchor discovery from cross-turn argument linking. Final price is evaluated by exact-match accuracy with no normalization or post-processing; trade outcome is evaluated as a three-way classification over Hit, Miss, and Unknown using accuracy and F1.

To assess the difficulty-aware router, we follow

RouteLLM (Ong et al., 2025) and report Performance Gap Recovery (PGR) and Call-Performance Threshold (CPT):

$$PGR(R^\alpha) = \frac{r(R^\alpha) - r(E_{\text{weak}})}{r(E_{\text{strong}}) - r(E_{\text{weak}})}, \quad (3)$$

where E_{weak} is the rule-based weak engine, E_{strong} is the LLM-powered Trade Engine with GPT-4o, and $r(\cdot)$ is response quality measured by accuracy. $CPT(x\%)$ is the minimum percentage of RFQ windows routed to E_{strong} needed to achieve $PGR = x\%$.

4.2 Results

We evaluate FinDialogLens along three dimensions aligned with our main claims: (1) whether message-level metadata and per-event RFQ windowing improve LLM-based role filling over full-chatroom CoT prompting; (2) whether fine-tuned open-source LLMs can approach proprietary LLM performance within the FinDialogLens pipeline; and (3) whether the difficulty-aware router can reduce LLM usage while preserving extraction accuracy. As a prerequisite, we verify that the Message-Level Module provides reliable metadata scaffolds for downstream windowing and role filling. The three specialized RoBERTa-base classifiers achieve F1 scores of 0.97, 0.95, and 0.93 for the RFQ Identifier, Trade Outcome Classifier, and Price NER model, respectively; per-model precision and recall are reported in Table 3.

4.2.1 Effect of Message-Level Metadata and Per-Event Windowing

Table 2 compares FinDialogLens with the Rule-based Engine and full-chatroom CoT-prompting

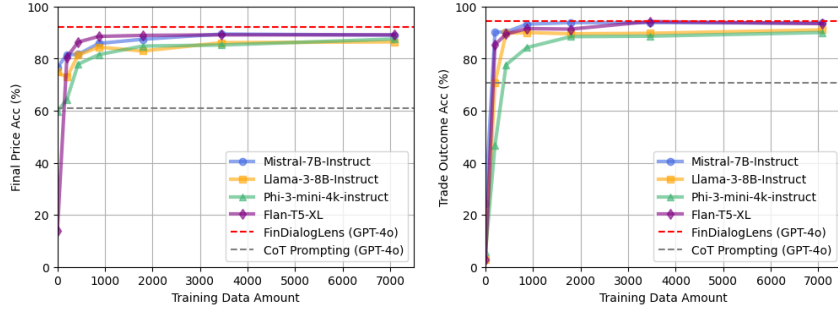


Figure 3: Accuracy of fine-tuned open-source LLMs on final price and trade outcome extraction as training-set size varies. Dashed lines mark FinDialogLens (GPT-4o) and zero-shot CoT Prompting (GPT-4o) baseline.

Model	Precision	Recall	F1
RFQ Identifier	0.97	0.97	0.97
Trade Outcome Classifier	0.96	0.93	0.95
Price NER Model	0.91	0.95	0.93

Table 3: Performance of the Message-Level Module, including the RFQ Identifier, Trade Outcome Classifier, and Price NER Model.

baselines. The “w/ Metadata” column denotes whether the input is augmented with predicted RFQ-trigger, price, and trade-outcome tags produced by the Message-Level Module. The CoT-prompting baselines apply GPT-4o to the full daily chatroom, whereas FinDialogLens first segments the chatroom into RFQ-specific windows and then performs role filling within each window. The main table reports one run for each method: GPT-4o-based methods use decoding with temperature set to 0, and fine-tuned open-source LLMs use greedy decoding. Because hosted API models can still exhibit residual nondeterminism even with temperature set to 0, Appendix C reports five-run mean and standard deviation for GPT-4o-based methods.

Message-level metadata substantially improves both the Rule-Based Engine and full-chatroom CoT prompting methods. For the Rule-Based Engine, message-level metadata enables final price and trade outcome accuracies of 77.1% and 90.3%, respectively. This improvement is largely due to the strong performance of the Message-Level Module, which helps the Rule-Based Engine link the correct final price and trade outcome to each detected RFQ trigger. However, the rule-based approach still struggles when multiple RFQs occur within a very short time span.

In the few-shot CoT setting, adding metadata increases final price accuracy from 64.4% to 79.2% and trade outcome accuracy from 70.5% to 87.0%,

corresponding to gains of 14.8 and 16.5 points, respectively. Similar improvements appear in the zero-shot setting, where metadata improves final price accuracy by 13.0 points and trade outcome accuracy by 20.4 points. These results show that compact message-level classifiers provide useful domain-specific signals for LLM-based extraction from noisy financial dialogue.

Per-event RFQ windowing provides a second, complementary gain. Compared with the strongest metadata-augmented CoT baseline, FinDialogLens (GPT-4o) improves final price accuracy from 79.2% to 92.1% and trade outcome accuracy from 87.0% to 94.3%, yielding gains of 12.9 and 7.3 points, respectively. This suggests that isolating each RFQ into a focused context window reduces cross-event interference and makes argument role filling easier for the LLM.

Overall, message-level metadata and per-event windowing jointly drive FinDialogLens’s improvement. Metadata provides compact domain-specific signals, while per-event windowing focuses the LLM on the relevant RFQ context for role filling. Together, these components allow FinDialogLens to achieve the best overall performance among all compared methods, with 92.1% final price accuracy and 94.3% trade outcome accuracy.

4.2.2 Open-source vs. Proprietary LLMs

Figure 3 shows that fine-tuned open-source LLMs in FinDialogLens improve rapidly on both final price and trade outcome extraction as the amount of in-domain training data increases, with performance plateauing after about 1,000 examples. With the full training set, all four open-source LLMs outperform GPT-4o CoT prompting baselines in Table 2. For instance, Flan-T5-XL achieves 91.7% final price accuracy and 92.8% trade outcome accuracy. These results show that modest in-domain

Router Type	Final Price		Trade Outcome	
	CPT (50%)	CPT (80%)	CPT (50%)	CPT (80%)
Random	49.2%	81.3%	52.1%	75.8%
RoBERTa-base	14.9%	32.4%	29.5%	62.6%
Mistral-7B-Instruct	16.4%	31.8%	26.7%	36.3%
Flan-T5-XL	16.1%	29.9%	25.1%	32.8%

Table 4: Call-Performance Threshold (CPT) for different router types. CPT is the fraction of RFQ windows routed to the LLM-powered Trade Engine required to recover a target percentage of the gap between the rule-based weak engine and FinDialogLens (GPT-4o). Bold indicates the best (lowest) value per column.

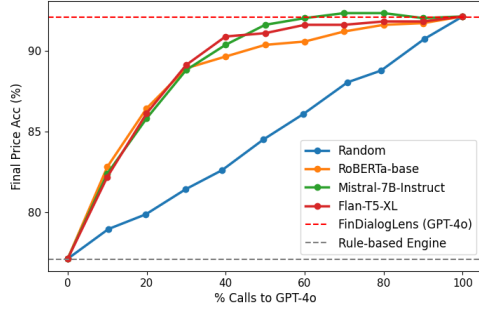


Figure 4: Performance of routers under various budget constraints (α).

supervision enables open-source LLMs with as few as 3B parameters to approach proprietary-model performance within the FinDialogLens pipeline.

4.2.3 Difficulty-Aware Router

The difficulty-aware router assigns each RFQ window to either the rule-based weak engine or the LLM-powered Trade Engine. The weak engine achieves 77.1% final price accuracy and 90.3% trade outcome accuracy on the test set, compared with 92.1% and 94.3% for FinDialogLens (GPT-4o), respectively. We evaluate routing by how efficiently it recovers this performance gap while reducing LLM calls, using both CPT and threshold-based analyses to measure the trade-off between accuracy and LLM usage.

Table 4 reports the Call-Performance Thresholds (CPT) for four routing strategies. Lower CPT indicates greater efficiency: fewer RFQ windows need to be routed to the LLM-powered Trade Engine to achieve a target PGR. Fine-tuned routers substantially outperform the Random baseline. For final price, RoBERTa-base reaches 50% PGR while routing only 14.9% of RFQ windows to the LLM-powered Trade Engine, and Flan-T5-XL reaches 80% PGR with 29.9%. For trade outcome, Flan-T5-XL requires only 25.1% and 32.8% of RFQ windows to reach 50% and 80% PGR, respectively.

Figure 4 shows the effect of varying α , which

adjusts the fraction of RFQ windows routed to the LLM-powered engine. Fine-tuned routers outperform the Random router in final price accuracy as the LLM call fraction decreases. Notably, they achieve performance close to running the LLM-powered engine on every RFQ window while routing only 40% of RFQ windows to it, indicating their effectiveness in assigning easier RFQ windows to the rule-based engine. Among the fine-tuned models, performance is similar, with no single model consistently superior.

Detailed routing-quality, cost, and latency analyses are in Appendix F. In our production setting of over 70,000 RFQs/day, routing 14.9% of RFQ windows to the LLM-powered Trade Engine for final price extraction recovers half of the accuracy gap between the weak engine and FinDialogLens (GPT-4o), saving over \$300/day in API spend.

5 Conclusion

We present **FinDialogLens**, a hybrid LLM pipeline for event extraction over multi-party dialogue, applied to missed-trade identification in financial chatrooms. Compact message-level classifiers provide inference-time metadata scaffolds, the RFQ-Level Module segments per-event RFQ windows, and the LLM-powered Trade Engine predicts the final price and trade outcome for each recovered RFQ. With GPT-4o, FinDialogLens reaches 92.1% and 94.3% accuracy on final price and trade outcome, outperforming full-chatroom CoT prompting; fine-tuned open-source LLMs with as few as 3B parameters achieve comparable performance with modest in-domain data. A difficulty-aware router further reduces cost by assigning easier RFQs to a rule-based weak engine and escalating harder RFQs to the LLM-powered Trade Engine, cutting LLM calls by 85% on final price while recovering half of the accuracy gap to FinDialogLens (GPT-4o), saving over \$300/day at our 70,000-RFQ/day scale.

Limitations

FinDialogLens provides a reusable framework for missed-trade identification in financial multi-party dialogue, a setting that has received limited attention in prior event extraction research. While the framework is designed to generalize beyond a single deployment environment, several components require adaptation before being applied to new institutions, asset classes, or communication settings.

First, FinDialogLens depends on high-performing compact models to generate message-level metadata, including RFQ-trigger labels, price mentions, and trade outcome cues. These signals are central to the pipeline: they guide RFQ window construction and provide inference-time scaffolding that helps the downstream LLM focus on relevant evidence. However, training reliable compact models requires labeled in-domain data, and their utility may decrease when annotations are scarce or when communication styles and product conventions shift. Second, the RFQ-Level Module currently constructs per-event RFQ windows using fixed temporal and message-count thresholds. Although these thresholds provide strong coverage in our setting, negotiation dynamics can vary substantially across desks, clients, and asset classes; some RFQs resolve immediately, while others unfold over longer intervals with clarification, requoting, or follow-up. Adaptive boundary detection conditioned on dialogue structure, participant behavior, and product-specific patterns could improve portability. Third, our difficulty-aware router is tailored to our deployment setting, where the primary decision is whether to use a low-cost rule-based engine or escalate to an LLM-powered Trade Engine. This binary design captures the cost–accuracy trade-off most relevant to our use case, but other deployments may require multi-way, cost-aware, or task-specific routing across multiple model tiers, specialist engines, latency constraints, or risk-based escalation policies.

Finally, due to client and regulatory confidentiality, the underlying conversations cannot be publicly released, limiting direct external replication on the original corpus. To partially mitigate this limitation, we provide annotation guidelines, dataset statistics, model architectures, training configurations, hyperparameters, and evaluation protocols to support reproduction on comparable proprietary financial dialogue datasets.

Ethics Statement

This research is driven by the goal of enhancing operational efficiency in financial markets through the automated analysis of unstructured communication channels, such as chatrooms and instant messaging platforms. The proposed framework, FinDialogLens, is designed to support financial professionals in identifying missed trading opportunities and improving decision-making, while complementing, rather than replacing, human expertise and judgment.

The financial data and conversational content analyzed in this study are highly sensitive and managed in accordance with strict industry governance and privacy protocols. All experiments were conducted using anonymized, internally governed datasets, with user identities replaced by generic identifiers to ensure privacy. No personal or client-identifying information was collected, stored, or utilized at any stage. Access to both data and model outputs is strictly restricted to authorized personnel, and all data handling complies with relevant regulatory and organizational standards.

The sample dialogues presented in this paper do not represent the opinions or positions of the authors or their organizations. We are committed to maintaining the highest standards of data privacy, security, and research integrity throughout this work.

Disclaimer

This paper was prepared for informational purposes in part by the Machine Learning Center of Excellence group of JPMorgan Chase & Co. and its affiliates (“JP Morgan”) and is not a product of the Research Department of JP Morgan. JP Morgan makes no representation and warranty whatsoever and disclaims all liability, for the completeness, accuracy or reliability of the information contained herein. This document is not intended as investment research or investment advice, or a recommendation, offer or solicitation for the purchase or sale of any security, financial instrument, financial product or service, or to be used in any way for evaluating the merits of participating in any transaction, and shall not constitute a solicitation under any jurisdiction or to any person, if such solicitation under such jurisdiction or to such person would be unlawful.

References

- Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav Chaudhary, Dong Chen, Dongdong Chen, and 110 others. 2024. [Phi-3 technical report: A highly capable language model locally on your phone](#). *Preprint*, arXiv:2404.14219.
- Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. [Event extraction via dynamic multi-pooling convolutional neural networks](#). In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 167–176, Beijing, China. Association for Computational Linguistics.
- Zhiyu Zoey Chen, Jing Ma, Xinlu Zhang, Nan Hao, An Yan, Armineh Nourbakhsh, Xianjun Yang, Julian McAuley, Linda Petzold, and William Yang Wang. 2024. [A survey on large language models for critical societal domains: Finance, healthcare, and law](#). *Preprint*, arXiv:2405.01769.
- Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, and 16 others. 2022. [Scaling instruction-finetuned language models](#). *Preprint*, arXiv:2210.11416.
- George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. [The automatic content extraction \(ACE\) program – tasks, data, and evaluation](#). In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC’04)*, Lisbon, Portugal. European Language Resources Association (ELRA).
- Seth Ebner, Patrick Xia, Ryan Culkin, Kyle Rawlins, and Benjamin Van Durme. 2020. [Multi-sentence argument linking](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8057–8077, Online. Association for Computational Linguistics.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. [The llama 3 herd of models](#). *Preprint*, arXiv:2407.21783.
- I-Hung Hsu, Kuan-Hao Huang, Elizabeth Boschee, Scott Miller, Prem Natarajan, Kai-Wei Chang, and Nanyun Peng. 2022. [DEGREE: A data-efficient generation-based event extraction model](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1890–1908, Seattle, United States. Association for Computational Linguistics.
- Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Léo Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. [Mistral 7b](#). *Preprint*, arXiv:2310.06825.
- Jonathan K. Kummerfeld, Sai R. Gouravajhala, Joseph J. Peper, Vignesh Athreya, Chulaka Gunasekara, Jatin Ganhotra, Siva Sankalp Patel, Lazaros C Polymenakos, and Walter Lasecki. 2019. [A large-scale corpus for conversation disentanglement](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3846–3856, Florence, Italy. Association for Computational Linguistics.
- Bobo Li, Hao Fei, Fei Li, Shengqiong Wu, Lizi Liao, Yinwei Wei, Tat-seng Chua, and Donghong Ji. 2024a. [Revisiting conversation discourse for dialogue disentanglement](#). *ACM Trans. Inf. Syst.*, 43(1).
- Qi Li, Heng Ji, and Liang Huang. 2013. [Joint event extraction via structured prediction with global features](#). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 73–82, Sofia, Bulgaria. Association for Computational Linguistics.
- Sha Li, Heng Ji, and Jiawei Han. 2021. [Document-level event argument extraction by conditional generation](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 894–908, Online. Association for Computational Linguistics.
- Yinheng Li, Shaofei Wang, Han Ding, and Hang Chen. 2024b. [Large language models in finance: A survey](#). *Preprint*, arXiv:2311.10723.
- Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. [Lost in the middle: How language models use long contexts](#). *Transactions of the Association for Computational Linguistics*, 12:157–173.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [Roberta: A robustly optimized bert pretraining approach](#). *Preprint*, arXiv:1907.11692.
- Yaojie Lu, Hongyu Lin, Jin Xu, Xianpei Han, Jialong Tang, Annan Li, Le Sun, Meng Liao, and Shaoyi Chen. 2021. [Text2Event: Controllable sequence-to-structure generation for end-to-end event extraction](#).

In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2795–2806, Online. Association for Computational Linguistics.

Yuqi Nie, Yaxuan Kong, Xiaowen Dong, John M. Mulvey, H. Vincent Poor, Qingsong Wen, and Stefan Zohren. 2024. [A survey of large language models for financial applications: Progress, prospects and challenges](#). *Preprint*, arXiv:2406.11903.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. 2025. [Routellm: Learning to route llms with preference data](#). *Preprint*, arXiv:2406.18665.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Madry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, and 401 others. 2024. [Gpt-4o system card](#). *Preprint*, arXiv:2410.21276.

Oscar Sainz, Iker García-Ferrero, Rodrigo Agerri, Oier Lopez de Lacalle, German Rigau, and Eneko Agirre. 2024. [Gollie: Annotation guidelines improve zero-shot information-extraction](#). *Preprint*, arXiv:2310.03668.

Soumya Sharma, Tapas Nayak, Arusarka Bose, Ajay Kumar Meena, Koustuv Dasgupta, Niloy Ganguly, and Pawan Goyal. 2023. [Finred: A dataset for relation extraction in financial domain](#). *Preprint*, arXiv:2306.03736.

Bo Wang, Heyan Huang, Xiaochi Wei, Ge Shi, Xiao Liu, Chong Feng, Tong Zhou, Shuaiqiang Wang, and Dawei Yin. 2023. [Boosting event extraction with denoised structure-to-text augmentation](#). In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11267–11281, Toronto, Canada. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS ’22*, Red Hook, NY, USA. Curran Associates Inc.

Hao Yu, Zachary Yang, Kellin Pelrine, Jean Francois Godbout, and Reihaneh Rabbany. 2023. Open, closed, or small language models for text classification? *arXiv preprint arXiv:2308.10092*.

Shun Zheng, Wei Cao, Wei Xu, and Jiang Bian. 2019. [Doc2EDAG: An end-to-end document-level framework for Chinese financial event extraction](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language*

Processing (EMNLP-IJCNLP), pages 337–346, Hong Kong, China. Association for Computational Linguistics.

A Data Privacy, Annotation Guidelines, and Statistics

Privacy: To ensure the privacy and confidentiality of conversations used in our framework, we implemented dedicated anonymization measures. Specifically, user names within the conversations are replaced with generic identifiers such as “Client_” and “Sales_,” effectively safeguarding individual identities.

Annotation Guidelines: To ensure the accuracy and reliability of our human-labeled data, we adopted a rigorous annotation protocol involving two professional annotators, each of whom independently reviewed every message and Request for Quote (RFQ). For message-level annotation, both annotators labeled the RFQ-triggering message, trade outcome classification, and price information associated with each message. For RFQ-level annotation, each annotator independently assigned the final price and trading outcome for each RFQ, with outcomes categorized as Hit, Miss, or Unknown.

To maintain annotation quality, we excluded cases falling into either of the following categories: (1) *Annotator-disagreement cases*: cases in which both annotators were able to assign labels based on the chat context, but their labels differed. (2) *Unresolvable cases*: cases in which the gold final price or trade outcome could not be determined from the chatroom alone because the negotiation moved to other communication channels, such as phone or email.

The fraction of cases discarded due to direct annotator disagreement was very small, accounting for less than 1% of the data. Because our annotators are professionals with domain experience, most labels were consistent across annotators. Separately, approximately 1% of RFQs were marked as unresolvable because the raw chatroom data did not contain sufficient information to determine the gold final price or trade outcome.

We provide several examples of these unresolvable cases below. In these examples, XYZ and ABC are placeholders used to mask product or security identifiers for confidentiality reasons.

Examples of Unresolvable Cases. Below are examples of RFQs that were marked as unresolvable.

Difficulty	Count	Percentage
Easy: 1 RFQ in window	598	61.9%
Hard: ≥ 2 RFQs in window	368	38.1%
Total	966	100.0%

Table 5: Distribution of test RFQs by the number of gold RFQ-trigger messages in the RFQ window. Hard cases correspond to interleaved RFQ windows containing two or more RFQs.

In these cases, the final price or trade outcome cannot be determined from the chat text alone because the negotiation moved to another communication channel. Therefore, we exclude these cases as out of scope.

Phone

<user_1>: Could you please price XYZ?
 <user_2>: sure
 <user_2>: -16.5 / -15
 <user_1>: calling you
 <user_2>: ok

Email

<user_1>: Please price XYZ
 <user_2>: can u send an email
 <user_2>: resending email confirm
 <user_1>: email sent

Statistics of Challenging Interleaved RFQs. To quantify the prevalence of challenging interleaved RFQs in the test set, we categorize each test RFQ according to the number of gold RFQ-trigger messages appearing within its RFQ window. In total, 368 out of 966 RFQs, corresponding to 38.1% of the test set, occur in windows containing two or more gold RFQ-trigger messages. These cases represent the type of interleaved RFQ scenarios illustrated in Figure 5, where multiple RFQs appear in close proximity and a model may incorrectly associate a price or trading outcome with the wrong RFQ.

B Training Details of FinDialogLens

All experiments are conducted with a fixed random seed of 42 and, where applicable, a temperature setting of 0. Model training is performed on a machine equipped with 4 NVIDIA A10G GPUs.

B.1 Message-Level Module

Inspired by recent work (Yu et al., 2023) demonstrating that fine-tuning compact models such as

RoBERTa with sufficient data can outperform larger models like GPT-4 in specialized tasks, we fine-tune three RoBERTa-base models within the Message-Level module to generate more sophisticated financial outputs for each message. Formally, for each message $m_i \in C$:

- **RFQ Identifier:** Determines whether a message m_i constitutes a Request for Quote (RFQ). We define a binary function:

$$y_i^{\text{RFQ}} = f_{\text{RFQ}}(m_i) \in \{\text{True}, \text{False}\} \quad (4)$$

where $y_i^{\text{RFQ}} = \text{True}$ indicates that m_i is identified as an RFQ message.

- **Trade Outcome Classifier:** Determines whether a message m_i indicates a trade outcome as either “Hit”, “Miss”, or “None.” We define:

$$y_i^{\text{trade}} = f_{\text{trade}}(m_i) \in \{\text{Hit}, \text{Miss}, \text{None}\} \quad (5)$$

- **Price NER Model:** Identifies and extracts the price entity from the message content. We define:

$$y_i^{\text{price}} = f_{\text{NER}}(m_i) \quad (6)$$

where y_i^{price} is the list of extracted price values from m_i .

The predictions y_i^{RFQ} , y_i^{trade} , and y_i^{price} are stored as metadata for each message m_i and are utilized in the subsequent module of our framework. The precision, recall, and F1 scores for these models are reported in Table 3.

B.2 Trade Engine

For the GPT-4o engine, we use “gpt-4o-2024-05-13” API version. We fine-tune four open-source LLMs as Trade Engines: (1) Mistral-7B-Instruct, (2) Llama-3-8B-Instruct, (3) Phi-3-mini-4k-instruct, and (4) Flan-T5-XL. The unified prompt template for all LLMs is shown in Figure 9. The training configuration is as follows:

1. Batch_size = 2
2. Gradient_accumulation_steps = 16
3. LoRAConfig = (lora_alpha = 16, lora_dropout = 0.1, r = 64)
4. Optimizer: AdamW

5. Epochs and learning rates (vary by training dataset size):

#Train	#Epoch	Learning rate
200	10	5e-5
435	10	5e-5
870	10	5e-5
1790	20	5e-5
3455	20	5e-5
7082	20	5e-5

B.3 Difficulty-Aware Router

In this section, we elaborate on the training details for three types of routers:

- Encoder-only Classifier:** RoBERTa-base is fully fine-tuned for this classifier. A classification layer is applied to the <s> token embedding to determine the class label L . The model is trained using cross-entropy loss, optimizing the parameters to accurately classify each input context window based on the provided labels.
- Decoder-only Classifier:** Mistral-7B-Instruct is fine-tuned using a prompt template to generate a single token, either “Yes” or “No.” A “Yes” indicates that the rule-based engine is correct on the input context window, while a “No” indicates incorrectness. The probability of correctness is computed using a softmax over “Yes” and “No,” with “Yes” corresponding to $P_{\theta}(\text{Correct} \mid w)$.
- Encoder-Decoder Classifier:** Flan-T5-XL is fine-tuned for this classifier under the same training and inference pipeline as the Decoder-only Classifier, with both encoder and decoder used to process the input context window and emit a single token, either “Yes” or “No”.

Below are the training configurations:

RoBERTa-base:

- Epoch = 5
- Learning_rate = 2e-5
- Batch_size = 32
- Gradient_accumulation_steps = 8
- Optimizer: AdamW

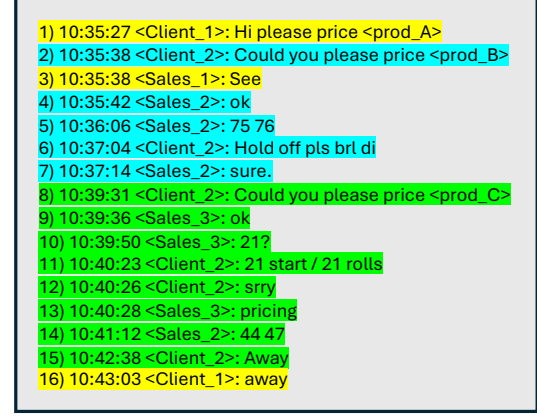


Figure 5: Example of an edge case in financial dialogue with three RFQs (messages 1, 2, and 8) occurring within a short time frame. Messages are highlighted in different colors to clarify which replies correspond to each RFQ event.

Mistral-7B-Instruct and Flan-T5-XL:

- Prompt template is shown in Figure 10
- Epoch = 5
- Learning_rate = 1e-4
- Batch_size = 4
- Gradient_accumulation_steps = 32
- LoRAConfig = (lora_alpha = 16, lora_dropout = 0.1, r = 64)
- Optimizer: AdamW

C Multi-Run Evaluation for GPT-4o-Based Methods

For GPT-4o-based methods, we use temperature-0 decoding. Although this setting is near-greedy, API-side nondeterminism can still introduce small run-to-run variation. We therefore run each GPT-4o-based setting five times and report the mean and standard deviation in Table 8.

D Error Analysis

We conducted error analysis of FinDialogLens (GPT-4o) on the test data and found that most edge cases occurred when multiple RFQs were requested within a short time period. In these complex scenarios, the LLM occasionally failed to correctly identify which response message corresponded to which RFQ.

Router Type	Final Price		Trade Outcome	
	CPT (50%)	CPT (80%)	CPT (50%)	CPT (80%)
Random	\$2.66	\$4.39	\$2.81	\$4.09
RoBERTa-base	\$0.80	\$1.75	\$1.59	\$3.38
Mistral-7B-Instruct	\$0.89	\$1.72	\$1.44	\$1.96
Flan-T5-XL	\$0.87	\$1.61	\$1.36	\$1.77

Table 6: Cost analysis of different router types for final price and trade outcome under CPT (50%) and CPT (80%). Reported values are USD per 1,000 RFQs; processing 1,000 RFQs with GPT-4o costs approximately \$5.40.

Router Type	Final Price		Trade Outcome	
	CPT (50%)	CPT (80%)	CPT (50%)	CPT (80%)
Random	717.8	1186.2	760.1	1105.9
RoBERTa-base	217.4	427.7	430.4	913.3
Mistral-7B-Instruct	239.3	464.0	389.6	529.6
Flan-T5-XL	234.9	436.2	366.2	478.6

Table 7: Latency analysis of different router types for final price and trade outcome under CPT (50%) and CPT (80%). Reported values are total LLM API latency (seconds) for the full test set; processing the full test set with GPT-4o alone takes approximately 1,459 seconds.

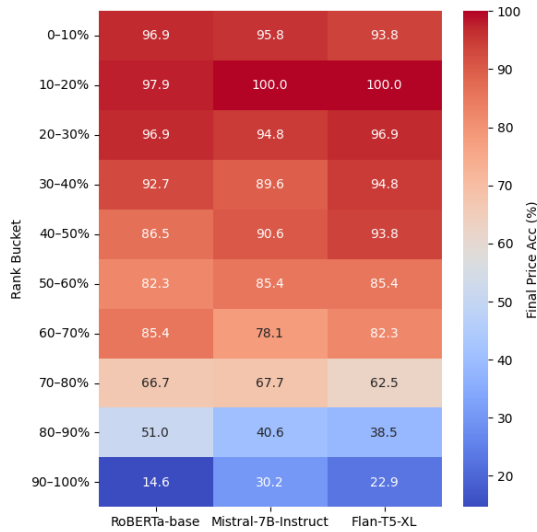


Figure 6: Accuracy of the rule-based engine on per-vent RFQ windows grouped into deciles by router logit ranking.

For example, in Figure 5, the target RFQ is the first message of the conversation, where <Client_1> requests a price for <prod_A>. Shortly after, <Client_2> submits a second RFQ for <prod_B> (message 2), and both RFQs receive prompt replies from <Sales_1> and <Sales_2>, respectively. Notably, <Sales_1> does not provide any price for the target RFQ, while <Sales_2> offers a price of “75 76” in message 5, which corresponds to the second RFQ. Eventually, <Client_1> replies “away”

in message 16, indicating the target RFQ was a missed trade. In this scenario, the LLM correctly identifies the target RFQ as a missed trade based on message 16, but incorrectly links the price “75 76” from message 5 to the target RFQ. The correct final price for the target RFQ should be NONE.

E Ablation Study on Metadata Error Propagation

To quantify the impact of metadata errors and the extent to which they propagate to the downstream RFQ-level prediction task, we conduct an additional ablation study. In this study, we replace the predicted message-level price and trade-outcome metadata with human-annotated gold labels, while keeping GPT-4o as the Trade Engine. This allows us to compare the original pipeline, which uses predicted metadata, against an oracle setting in which the downstream Trade Engine receives gold message-level metadata.

The results suggest that error propagation from the message-level metadata module is limited. Replacing predicted metadata with gold metadata changes final price accuracy only marginally, from 92.1% to 91.9%, while improving trade outcome accuracy from 94.3% to 95.9% and trade outcome F1 from 94.5% to 95.7%. These results indicate that the downstream Trade Engine is relatively robust to the remaining errors produced by the high-

Method	w/ Metadata	Input Context		Trade Engine	Final Price	Trade Outcome	
		Whole Day	RFQ Windows		Acc	Acc	F1
CoT Prompting	✓	✓		GPT-4o (Zero-shot)	62.2 $\pm$ 0.95	66.7 $\pm$ 0.94	76.7 $\pm$ 0.37
		✓		GPT-4o (Few-shot)	66.1 $\pm$ 2.17	67.5 $\pm$ 2.47	77.2 $\pm$ 2.09
		✓		GPT-4o (Zero-shot)	76.4 $\pm$ 1.73	84.4 $\pm$ 1.32	89.7 $\pm$ 1.40
		✓		GPT-4o (Few-shot)	79.2 $\pm$ 0.59	85.2 $\pm$ 0.81	90.3 $\pm$ 0.93
FinDialogLens	✓		✓	GPT-4o	91.8 $\pm$ 0.42	93.4 $\pm$ 0.27	93.5 $\pm$ 0.38

Table 8: Multi-run results for GPT-4o-based methods. We report mean accuracy/F1 over five runs, with standard deviation in subscript. All GPT-4o runs use temperature-0 decoding.

Method	Final Price	Trade Outcome	
	Acc	Acc	F1
FinDialogLens (GPT-4o)	92.1	94.3	94.5
FinDialogLens (GPT-4o) + oracle message-level metadata	91.9	95.9	95.7

Table 9: Ablation study on metadata error propagation. We compare the original FinDialogLens pipeline using predicted message-level metadata with an oracle setting that replaces predicted metadata with gold message-level labels.

performing Message-Level Module.

F Difficulty-Aware Router Analysis

F.1 Routing Quality

To further assess routing quality, we ranked all RFQ windows by their router logits and partitioned them into ten equal-sized bins. We then evaluated the accuracy of the rule-based engine within each bin to analyze the relationship between router confidence and routing effectiveness. As shown in Figure 6, the top-ranked bins (0–10%, 10–20%, 20–30%) achieved outstanding performance, with final price prediction accuracy ranging from 95% to 100%. In contrast, the lowest-ranked bin (90–100%) exhibited a markedly lower accuracy of just 14.6% in the RoBERTa-base router. These findings indicate that our fine-tuned routers are highly effective at identifying which RFQ events are most appropriate for rule-based processing.

F.2 Cost and Latency

We provide a detailed analysis of cost and latency for the routing experiments. The average cost per RFQ using GPT-4o is \$0.0054, with an average of 1,024.4 input tokens and 18.5 output tokens per RFQ. As shown in Table 6, processing over 70,000 RFQs per day with the RoBERTa-base router results in daily savings exceeding \$300, while maintaining approximately 85% accuracy for final price extraction.

For latency, the average response time for a single GPT-4o API call is 1.51 seconds per RFQ event. In our latency experiments, we measured the total processing time by sequentially submitting one RFQ event at a time, without batching or parallelization. The total latency for processing the entire test set with GPT-4o is approximately 1,459 seconds.

Table 7 reports total LLM API latency (in seconds) for the different router types under CPT(50%) and CPT(80%). Fine-tuned routers such as RoBERTa-base and Flan-T5-XL substantially reduce overall latency compared with the Random baseline. Under CPT(50%), the RoBERTa-base router reduces total latency for final-price extraction to 217.4 seconds, compared with 717.8 seconds for the Random router. Flan-T5-XL achieves 234.9 seconds under CPT(50%) for final price and 366.2 seconds for trade outcome, both well below the Random router’s 760.1 seconds. These reductions support the practicality of our framework in deployment.

- 1) 09:12:00 Client_1:Pls price 70mm 0.375 9/27 in 5y box <<rfq>>
- 2) 09:12:10 Sales_1: sure
- 3) 09:12:45 Client_1: Also price 100mm 3.5\1/28 in box both can trade <<rfq>>
- 4) 09:14:07 Sales_1: 9/27: -244/-27 <<price>>
- 5) 09:14:25 Sales_1: 1/28: -29/-30 <<price>>
- 6) 09:15:00 Client_1: Can you tighten the spread on 9/27?
- 7) 09:15:10 Sales_1: 9/27: -242/-27 <<price>>
- 8) 09:15:36 Client_1: Ntg to do both thanks <<miss>>
- 9) 13:10:11 Client_2: Pls quote 40 ES calls <<rfq>>
- 10) 13:13:56 Sales_2: 2.15/2.25 <<price>>
- 11) 13:15:43 Client_2: done <<hit>>

1) 09:12:00 Client_1: Pls price 70mm 0.375 9/27 in 5y box
2) 09:12:10 Sales_1: sure
3) 09:12:45 Client_1: Also price 100mm 3.5\1/28 in box both can trade
4) 09:14:07 Sales_1: 9/27: -244/-27 <<price>>
5) 09:14:25 Sales_1: 1/28: -29/-30 <<price>>
6) 09:15:00 Client_1: Can you tighten the spread on 9/27?
7) 09:15:10 Sales_1: 9/27: -242/-27 <<price>>
8) 09:15:36 Client_1: Ntg to do both thanks
Potential Hit: []
Potential Miss: ["09:15:36 Client_1: Ntg to do both thanks"]

1) 09:12:45 Client_1: Also price 100mm 3.5\1/28 in box both can trade
2) 09:14:07 Sales_1: 9/27: -244/-27 <<price>>
3) 09:14:25 Sales_1: 1/28: -29/-30 <<price>>
4) 09:15:00 Client_1: Can you tighten the spread on 9/27?
5) 09:15:10 Sales_1: 9/27: -242/-27 <<price>>
6) 09:15:36 Client_1: Ntg to do both thanks
Potential Hit: []
Potential Miss: ["09:15:36 Client_1: Ntg to do both thanks"]

1) 13:10:11 Client_2: Pls quote 40 ES calls
2) 13:13:56 Sales_2: 2.15/2.25 <<price>>
3) 13:15:43 Client_2: done
Potential Hit: ["13:15:43 Client_2: done"]
Potential Miss: []

Task

[Output Format: describe the output format]

You are a smart financial assistant designed to provide final price, and trade outcome for the targeted RFQ (Request for Quote) appeared in the conversation.

Background Knowledge

[Provide a brief introduction to RFQs, including what an RFQ is and how clients and sales teams negotiate prices.]

Tips

[Include step-by-step Chain-of-Thought guidelines for RFQ extraction.]

Conversation

{Input Content}

Special tokens

[Describe the meaning of each special token, e.g., <<rfq>>, <<hit>>, <<miss>>, and <<price>>.]

Task

[Output Format: describe the output format]

Figure 9: Prompt template for the Trade Engine within FinDialogLens.

You are a helpful evaluator. Given a conversation, your task is to determine whether the rule-based system can successfully detect the final price and trading outcome for the target RFQ.

Background Knowledge

[Provide a brief introduction to RFQs, including what an RFQ is and how clients and sales teams negotiate prices.]

Rule-based System

[Provide the pre-defined rules used in the rule-based system.]

Conversation

{Input Content}

Task

[Output Format: describe the output format]

Figure 10: Prompt template for the difficulty-aware router.