EvoTac: A Self-Evolving LLM Agent for Eliciting Reusable Tacit Negotiation Heuristics from Terminal Outcomes

Anonymous ACL submission

Abstract

We propose EvoTac, an LLM-based framework for real-world negotiation that converts sparse terminal outcomes into reusable tacit experience without fine-tuning the base model. It continuously adapts to changing opponents and scenarios through a simple predict–reflect–update loop, using decoupled layered memory to represent the agent's constraints, observed opponent behavior patterns, and persistent hypotheses about opponent stance/type. Experiments on a real-world online marketing negotiation task (predicting final commission rates) show that EvoTac outperforms traditional models and multiple LLM baselines in prediction accuracy and first-round offer hit rate.

1 Introduction

In complex strategic settings such as climate policy (Liu et al., 2025), financial markets (Bajari et al., 2007), and the internet economy (Edelman et al., 2007; Varian, 2007; Rong et al., 2015), outcomes hinge on whether agents can form mutually consistent expectations and settle on stable, self-consistent behavior. However, in real negotiations, what enables such stability is often not a one-off "best" strategy, but a set of *reusable, situation-conditioned heuristics* distilled from experience: what terms the counterparty can accept, when they concede, what triggers regime shifts, and which trade-offs are consistently valued. Turning observed negotiation trajectories and terminal outcomes into such transferable heuristics is therefore a practical capability beyond equilibrium computation: it supports policy coordination, risk pricing, market design, and platform governance, and enables negotiation systems to generalize across high-dimensional, non-stationary environments.

Existing approaches—classical numerical methods, supervised learning, and multi-agent reinforcement learning (MARL)—can work well in structured settings, but in real-world negotiations they often fall short in (i) capturing tacit strategic regularities that are only revealed through delayed outcomes, (ii) providing interpretable rationales that can be *reused* as decision heuristics, and (iii) transferring effectively across changing or out-of-distribution scenarios. In particular, many learning-based methods ultimately output a policy or a prediction, while leaving the core question unanswered: *what reusable strategic heuristics have been learned from outcomes, and how are they updated when outcomes contradict expectations?*

Large language models (LLMs) open a new route to interpretable strategic modeling: they can express decision rationales in natural language and may exhibit robustness under distribution shift. Prior work supports this promise in complex games: Cicero combines language-based negotiation with strategic reasoning to produce interpretable decisions (Bakhtin et al., 2022), and DipLLM (Xu et al., 2025) decomposes decision-making in Diplomacy into unit-level serialized action selection and iteratively generates strategies via next-token prediction. Yet most existing approaches still treat LLMs primarily as *strategy generators* or static knowledge bases, lacking an endogenous mechanism that uses sparse, delayed terminal outcomes to extract, validate, and refine the underlying *heuristic-like representations* that drive stable agreements.

We observe that, in real negotiations, the key drivers of stable outcomes are often not explicit rules but tacit heuristics that can be repeatedly validated and corrected through experience. Negotiation failures frequently stem from overlooking or misestimating latent counterparty-side factors—for example, the opponent's **stance**, an evolving **acceptable range**, **deadline sensitivity** under pressure, **trade-off valuation** across price and non-price clauses, and **switching conditions** that trigger abrupt regime shifts or exit. Without sufficient tacit heuristics about the opponent, proposed terms

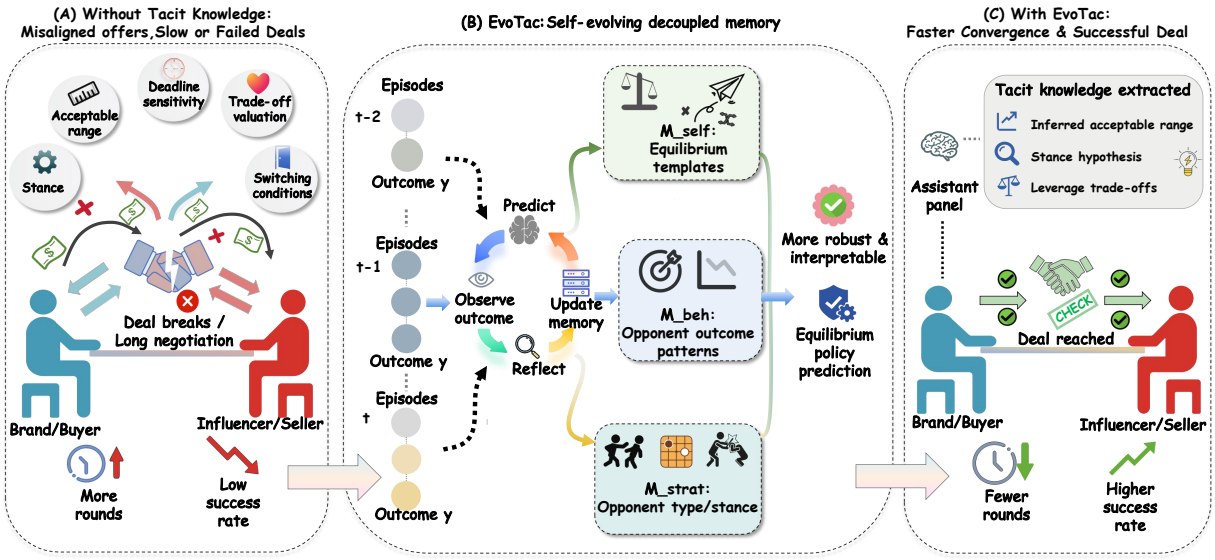

Figure 1: EvoTac motivation: why tacit knowledge matters for negotiation & heuristic elicitation from outcomes.

can be systematically misaligned with what the opponent can actually accept, leading to slow convergence or failed deals as shown in Figure 1 (A). In contrast, if an agent can reflect on deviations between *expected* and *realized* outcomes and continually update its internal heuristic set, it can better anticipate equilibrium-like reactions, adapt to shifts in counterparties, and reach agreements more efficiently.

Motivated by this observation, we propose EvoTac: a cognitively inspired, LLM-based framework that continuously *elicits and evolves reusable social tacit heuristics* from terminal feedback in real negotiation settings. Importantly, EvoTac is not designed to merely predict a strategy or an outcome. Instead, its primary objective is to construct a *computable, interpretable, and updateable heuristic base* that summarizes what has been learned from real negotiation outcomes. Predicting equilibrium-like strategies or terminal results is treated as a *downstream application*—a way to operationalize and empirically validate whether the elicited heuristics are correct, reusable, and adaptive.

Concretely, EvoTac comprises three tightly coupled modules that form a self-evolving *predict–reflect–update* loop across offline and online phases (Gao et al., 2025; Lin et al., 2025; Luo et al., 2025). A *self-evolving decoupled memory module* explicitly stores tacit heuristics that shape negotiation outcomes, organizing long-term information about self strategy, opponent behavior, and opponent types into a multi-layer structure. Building on this, an *experience reflection module* aligns the agent's heuristic-implied expectations with realized terminal feedback after each negotiation, leveraging the model's reasoning traces to diagnose *which heuristic assumptions failed and why*. The resulting diagnostic signals are then consumed by a *memory management module*, which performs structured updates over the multi-layer heuristic memories via atomic operations such as creation, reinforcement, correction, merging, and retirement, thereby enabling continual adaptation.

We evaluate EvoTac in a large-scale real-world online marketing scenario. To provide a measurable downstream task, we instantiate evaluation as a regression problem over the **final commission rate** negotiated between brands and influencer promoters. Using the same base LLM, we compare against multiple baselines, including traditional supervised learning models, long-context LLMs without explicit memory, LLM agents with single-layer memory or standard table-based retrieval/RAG, and an EvoTac variant with memory constructed only offline. Experimental results demonstrate that **by explicitly modeling and dynamically evolving decoupled multi-layer memories as reusable tacit heuristics**, language models can acquire a computable form of social tacit knowledge from terminal outcomes without updating base parameters. This heuristic-centric representation yields interpretable and robust equilibrium-like recommendations, improves negotiation efficiency and transaction success rates, and provides a methodological foundation for building negotiation agents with so-

cial competence and continual learning capabilities.

2 Methods

In this section, we formalize the learning setting and notation for evolving reusable tacit negotiation heuristics from real-world terminal outcomes. We then present the overall EvoTac loop (predict–reflect–update) and describe each module block by block.

2.1 Task Definition

We study *evolving reusable tacit negotiation heuristics* under realistic observability constraints: a strategy-proposing side can access partial profiles of both parties and the game context, but can learn only from historical *terminal outcomes* rather than full interaction traces.

Formally, each negotiation is a prediction instance. For the t-th instance, the input is

$$x_t = \langle \rho_t^{(A)}, \rho_t^{(B)}, \mathcal{C}_t \rangle,$$

where $\rho_t^{(A)}$ and $\rho_t^{(B)}$ denote the two parties' profile information, which can be natural-language summaries or structured fields (e.g., role types, historical styles, capability constraints, and preference/constraint cues). $\mathcal{C}_t$ denotes the game-context description of this negotiation, including the industry/channel, stage, institutional and resource constraints, external environment, and rule specifications.

The historical dataset consists of pairs (x_t, y_t), where y_t is the observed terminal settlement after real multi-round interaction. Our objective is to evolve a computable and interpretable heuristic base M from terminal outcomes only, and revise it when outcomes contradict heuristic-implied expectations. Given x_t and the current M, the agent outputs a heuristic-grounded recommendation $\hat{y}_t$ (or a strategy description) and a rationale s_t used for reflection and targeted heuristic updates once y_t is observed.

2.2 EvoTac Framework

2.2.1 Overall Workflow

EvoTac comprises three tightly coupled core modules: (i) a **self-evolving decoupled memory** module, which separately represents "self strategy boundaries/response structure—opponent behavioral regularities and drift—opponent type/stance and stable response structure"; (ii) an **Experience reflector**, which, after the terminal outcome becomes available, aligns the prediction with realized feedback and diagnoses the source of deviation using the current reasoning trace and retrieved evidence, producing executable minimal-change recommendations; and (iii) a **Memory manager**, which operationalizes the reflection signals into create, reinforce, update, merge, and deprecate operations in memory space.

By chaining these components, EvoTac forms a complete self-evolutionary loop over an online data stream. For each newly arriving instance x_t, the system first retrieves the most relevant tacit knowledge from multi-layer memory, constructs a structured prompt, and uses the LLM to produce a prediction $\hat{y}_t$ and an explanation s_t. When the terminal settlement y_t becomes available, the system performs reflective diagnosis and attribution on the deviation $|\hat{y}_t - y_t|$, and converts the diagnosis into "add/delete/modify" operations on memory units. The updated memory then immediately changes retrieval evidence and reasoning trajectories for the next instance, forming a cross-instance predict–reflect–update loop. In this way, the system can continuously distill, correct, and strengthen tacit knowledge on an online stream, realizing a self-evolutionary process of continual improvement. Concretely, *prediction*, *terminal-feedback diagnosis*, and *memory rewriting* are each implemented as prompt-driven LLM calls (same backbone, different role instructions); see Appendix §C for the instance-level prompt templates and output schemas.

2.2.2 Self-Evolving Decoupled Memory

As shown in Fig. 2 (A), stable negotiation outcomes are often driven by repeatedly validated **tacit experience** rather than explicit rules. Inspired by the "type–belief–strategy separation" in incomplete-information games, we organize memory into three decoupled layers: self strategy boundaries, opponent behavioral drift, and opponent type/stance hypotheses. Formally, the memory bank is

$$M = \{M_{\text{self}}, M_{\text{beh}}, M_{\text{strat}}\} \tag{1}$$

Self-strategy memory (M_{self}) characterizes our stable objectives, constraints, and decision boundaries under a given situation, and organizes them as executable response templates. It captures preconditions (e.g., brand tier and campaign goals that determine the feasible region), strategy templates (including anchor/first-offer heuristics, con-

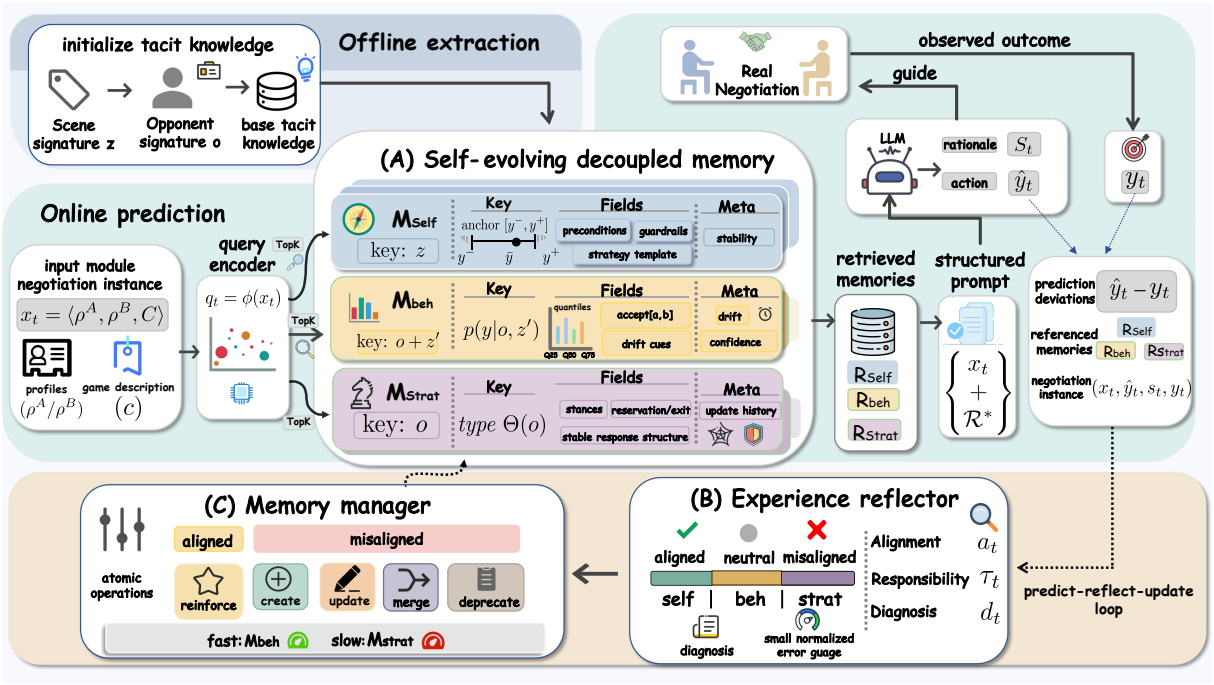

Figure 2: EvoTac framework.

cession schedules, non-price levers and triggers), and guardrails (such as min/max commission, ROI floor, and risk limits). Intuitively, it provides "the strategy boundaries and anchors for our side in this game scenario".

Opponent-behavior memory (M_{beh}) records short-term empirical regularities and drift in the opponent's observable outcome-space behavior. It maintains outcome statistics including recent accept ranges and quantiles (mean/std or quantile distributions), as well as drift cues that capture short-term shifts (e.g., upward/downward moves). This layer enables the system to quickly absorb "what has recently changed" for the opponent.

Opponent-strategy memory (M_{strat}) maintains slowly varying hypotheses about the opponent's *type and stance*, characterizing more stable response structures across scenarios. It captures stance hypotheses (stance labels and supporting cues), stable response structures (including anchor aggressiveness, concession speed, deadline/constraint sensitivity, and switching conditions), and reservation/exit information (acceptable ranges and termination/hardline triggers). By aggregating cross-scene evidence, this layer avoids being dominated by single-instance noise.

This "type/behavior/boundary" decoupling is beneficial because, when predictions are misaligned, it allows us to decompose error sources into (i) inappropriate self templates; (ii) uncaptured short-term behavioral drift; and (iii) biased type/stance hypotheses, thereby enabling **targeted, auditable, minimal-change** updates under terminal-only feedback. Each memory unit is stored as a structure comprising an index key (key), structured fields (value), and statistical/confidence metadata (meta), thereby enabling auditable and locally rollback-able updates. The field schema is shown in Table 4.

Offline Extraction: Initialize Tacit Knowledge

Before online prediction begins, we initialize the three-layer memory using two types of offline extraction from historical negotiation data.

(1) **Scene-prototype and anchor extraction**: based on the context $\mathcal{C}_i$ (industry/stage/constraints/rules, etc.), we cluster or match recurring situation prototypes, and within each prototype, compute robust statistics of terminal settlements y (e.g., quantile intervals and conditional means). These statistics are used to initialize anchors and response templates in M_{self}.

(2) **Opponent acceptance-range and type-prior extraction**: we aggregate terminal settlements by opponent identity or identifiable features to estimate empirical acceptance ranges, drift statistics, and uncertainty, populating M_{beh}. Meanwhile, we extract slower-varying type priors from cross-context consistency signals to initialize M_{strat}.

This procedure constructs a retrievable tacit-

knowledge base, allowing online inference to start from reasonable anchors and to be continuously corrected via subsequent reflection.

Multi-Layer Memory Retrieval

At online prediction, we retrieve the K most relevant memories from each layer using similarity search based on the current input x_t. The retrieved triplets are compressed and concatenated with x_t into a structured prompt for the LLM, which outputs prediction $\hat{y}_t$ and explanation s_t.

2.3 Experience Reflector: From Error to Alignment Signals

Prediction $\hat{y}_t$ and explanation s_t are used as heuristic-grounded recommendations that guide one party engaged in the real negotiation. After the real negotiation concludes, the terminal settlement y_t is observed.

As shown in Fig. 2 (B), after observing the terminal settlement y_t, the Experience reflector translates *prediction deviations* into diagnostic signals for the Memory manager. It outputs: (1) a global alignment label a_t (aligned/neutral/misaligned); (2) a responsibility vector $\mathbf{r}_t$ that assigns credit/blame across the three memory layers; and (3) a textual diagnosis d_t that provides minimal-change recommendations, specifying which layer to revise, which fields to modify, and in what direction.

Alignment between the prediction $\hat{y}_t$ and the terminal outcome y_t is determined by scenario-normalized error. Let $[l_t, u_t]$ be the feasible interval induced by game description constraints and retrieved M_{self}:

$$e_t = \hat{y}_t - y_t, \qquad \epsilon_t = \frac{|e_t|}{u_t - l_t + \eta}, \tag{2}$$

where $\eta > 0$ is a numerical stabilizer to prevent the ratio from exploding under extremely narrow intervals. We then define the global alignment label

$$a_t = \begin{cases} \text{aligned}, & \epsilon_t \le \delta_1, \\ \text{misaligned}, & \epsilon_t \ge \delta_2, \\ \text{neutral}, & \delta_1 < \epsilon_t < \delta_2, \end{cases} \tag{3}$$

where $0 < \delta_1 < \delta_2 < 1$ are thresholds for "acceptable deviation" and "significant deviation," respectively. Under the same scenario-scale interval, predictions sufficiently close to the realized outcome are considered aligned; sufficiently large deviations are considered misaligned; and intermediate cases remain neutral to suppress over-updating.

The **responsibility vector** is generated via post-mortem reflection: given $(x_t, \hat{y}_t, s_t, y_t)$ and the memory snippets referenced by the current prediction, the LLM produces root-cause scores $\mathbf{s}_t \in \mathbb{R}^3$ (corresponding to $M_{\text{self}}, M_{\text{beh}}, M_{\text{strat}}$), which are normalized as $\mathbf{r}_t = \text{softmax}(\mathbf{s}_t)$. When a_t = misaligned, $\mathbf{r}_t$ concentrates blame on the most likely faulty layer; when a_t = aligned, it assigns credit to the hit layers and supports reinforcement. The Reflector thus outputs $(a_t, \mathbf{r}_t, d_t)$ to guide memory updates.

2.4 Memory Manager: Strategy Iteration in Memory Space

As shown in Fig. 2 (C), the Memory manager translates $(a_t, \mathbf{r}_t, d_t)$ into atomic operations on $(M_{\text{self}}, M_{\text{beh}}, M_{\text{strat}})$: create, reinforce, update, merge, and deprecate. Its core principle is **local rewriting, semantic alignment, and layer-wise update pacing**: short-term opponent behavior changes are absorbed quickly in M_{beh}, while M_{self} and M_{strat} emphasize cross-instance evidence to avoid disrupting stable structures.

Three cases drive the update behavior:

Aligned (a_t = aligned): Reinforce hit units in layers with high $\mathbf{r}_t$ weights (e.g., increase confidence, update success counts and timestamps, expand reusability tags), turning successful cases into stronger tacit knowledge that is more likely to be retrieved later.

Misaligned (a_t = misaligned): Treat $\mathbf{r}_t$ as an update budget. For the layer with maximal weight, rewrite hit units with minimal changes according to diagnosis d_t: e.g., adjust anchors/template parameters in M_{self}, update quantile intervals and drift statistics in M_{beh}, or revise type hypotheses and cross-context evidence aggregation in M_{strat}. Create new entries only when the diagnosis identifies a critical gap (i.e., no explainable entry exists in the current layer). When entries are highly similar or obsolete, trigger merge/deprecate.

Neutral (a_t = neutral): Perform no structural updates; only accumulate metadata (counts, timestamps, lightweight confidence) to avoid overfitting to noise. Explicit rewrites are triggered only after repeated neutral deviations accumulate in a consistent direction and cross-instance evidence exceeds thresholds.

For example, if $\hat{y}_t = 15\%$ but $y_t = 17\%$ and the instance is misaligned, d_t may indicate an upward shift in the opponent's acceptance interval. The Memory manager would prioritize updating

that opponent's acceptance range and drift statistics in M_{beh}, conservatively revise M_{strat} (unless cross-context evidence suggests a type/stance change), and only backtrack to revise generic templates in M_{self} if multiple scenario prototypes exhibit systematic bias. In this way, the system achieves continual self-evolution through explicit "retrieve–diagnose–rewrite" memory updates.

3 Experiment

We evaluate EvoTac in a brand-affiliate matchmaking workflow on a large internet marketing platform. We ask whether EvoTac can convert sparse *terminal* outcomes into reusable tacit negotiation heuristics, and whether these heuristics remain useful when counterparties and market conditions drift under partial observability.

As a measurable downstream validation task, we cast evaluation as commission anchoring: given pre-deal observable signals (brand/affiliate profiles, product and audience tags, historical performance) and platform statistics, predict the realized **final commission** y_{pp} measured in **percentage points (pp)** (e.g., $17\% \equiv 17$ pp). EvoTac is not involved in live bargaining and does not observe intermediate offers; it produces a heuristic-grounded quote (with a short rationale) as an actionable first-offer recommendation, and updates its memory only after the terminal outcome is observed. For negotiations that end without agreement, we treat the terminal result as a failure signal for diagnosis and memory revision; regression metrics are computed on instances with realized final rates.

We use a dataset from this platform covering commercial matchmaking in **calendar year 2024**, containing both **successful deals** and **failed negotiations**. Concretely, we use 500 human-audited matches with a chronological 70/30 split: 350 instances as the **offline set** for training baselines and initializing EvoTac's memory, and 150 instances as a **time-ordered online stream**. Baselines are trained only on the offline set and evaluated by single-pass prediction on the online set, whereas EvoTac performs online memory updates through a "predict–reflect–update" loop under **terminal-only supervision**.

3.1 Evaluation Metrics

We report standard regression quality together with a business-oriented hit rate (MAE/RMSE are reported in pp):

System	Method summary
XGB-Feat	XGBoost regressor on handcrafted/statistical features from profiles and transaction logs.
LLM-MonoMem	LLM agent with a single-layer memory pool (flattened from $M_{\text{self}}/M_{\text{beh}}/M_{\text{strat}}$), all entries share one schema and are jointly retrieved without layer decoupling.
LLM-LongCtx	One-shot long-context prediction by concatenating a bounded slice of offline history (up to 8k tokens).
LLM-TabRAG	k-NN retrieval over embedded profile/log rows; inject retrieved raw fields as tabular evidence for the LLM.
MARL	Two-agent self-play (brand/affiliate) with Q-learning in a simplified bargaining game: actions are discretized commission quotes and terminal rewards are computed from the observed settlement; the learned policy outputs the first-offer anchor.
EvoTac (off)	EvoTac with three-layer memory initialized from the offline set; memory is frozen during the online stream.
EvoTac (on)	Full EvoTac with terminal-feedback-driven diagnostics and controlled memory edits (create/reinforce/update/merge/deprecate).

Table 1: Overview of baselines.

- **MAE / RMSE (pp)**: mean absolute error and root mean squared error;
- **MAPE**: mean absolute percentage error;
- R^2: coefficient of determination;
- **Success@0.5pp**: fraction of predictions within ± 0.5 percentage points of the true commission (a practical "first-offer hit" tolerance);

3.2 Baseline Setting

We compare (i) feature-based regression, (ii) LLM predictors using long context or retrieval/memory augmentation, and (iii) MARL self-play trained from terminal rewards. EvoTac (off/on) isolates the two design choices emphasized in this work: decoupling heterogeneous tacit heuristics into layers, and rewriting these heuristics online from outcome-only feedback. For a fair comparison, all LLM-based methods share the same backbone (DeepSeek-v3.1), prompting style, and sampling policy (10 runs per input; median reported), with no parameter fine-tuning. Table 1 summarizes the seven systems.

3.3 Main Results

3.3.1 Overview

We evaluate EvoTac via three questions aligned with our goal of extracting and maintaining reusable tacit heuristics from outcomes:

- **RQ1**: Under terminal-only feedback, does EvoTac produce better commission anchors than non-evolving predictors (e.g., feature regression and MARL)?

System	MAE ↓	RMSE ↓	MAPE ↓	R^2 ↑	Success@0.5pp ↑
XGB-Feat	0.6002	0.7736	0.1004	0.7327	0.52
MARL	1.0734	1.2167	0.1907	0.3390	0.20
LLM-MonoMem	0.5796	0.8026	0.0933	0.7124	0.56
LLM-LongCtx	0.5986	0.7649	0.0967	0.7437	0.49
LLM-TabRAG	0.5895	0.7551	0.0997	0.7454	0.62
EvoTac (off)	0.4468	0.5621	0.0744	0.8589	0.68
EvoTac (on)	**0.4151**	**0.5084**	**0.0684**	**0.8846**	**0.70**

Table 2: Performance on the platform dataset (MAE/RMSE in pp).

- **RQ2**: Does the three-layer decoupled memory structure ($M_{\text{self}}/M_{\text{beh}}/M_{\text{strat}}$) improve the use of tacit regularities compared to single-layer memory, long-context prompting, or naive RAG?
- **RQ3**: Does online memory rewriting lead to sustained improvements in terminal outcomes relative to a frozen, offline-initialized heuristic base?

As shown in Table 2, EvoTac (on) performs best on all reported metrics, and online rewriting provides consistent gains over the frozen variant.

3.3.2 RQ1 Analysis: Gains over Non-Evolving Baselines

EvoTac secures better commission anchoring by **transforming outcome supervision into reusable tacit heuristics**. As detailed in Table 2, EvoTac (on) significantly reduces MAE from 0.6002 (XGB-Feat) to 0.4151 and improves the first-offer hit rate (Success@0.5pp) from 0.52 to 0.70, far surpassing the MARL baseline (0.20). These gains confirm that explicitly extracting transferable structures—such as feasibility guardrails, acceptance drift, and stance regularities—yields significantly better-calibrated anchors across heterogeneous opponents than static feature regression or prompt-based LLM baselines (LLM-MonoMem/LongCtx/TabRAG).

3.3.3 RQ2 Analysis: Efficacy of Decoupled Memory

To isolate the contribution of the structured memory, we compare EvoTac (off) against LLM baselines with identical backbones and prompts. EvoTac (off) already improves markedly (e.g., MAE 0.4468 vs. 0.5895 for LLM-TabRAG; R^2 0.8589 vs. 0.7454). The gap to LLM-MonoMem (MAE 0.5796) is consistent with the benefit of keeping heterogeneous tacit signals separable: feasibility-aligned anchors and guardrails (M_{self}), short-horizon acceptance dynamics and drift cues (M_{beh}), and slower stance/type structure (M_{strat}) are retrieved jointly but written and revised independently. This separation makes misses attributable to a specific layer and enables localized edits, which is especially useful when intermediate dialogue traces are unobserved.

Variant	MAE	RMSE	R^2	Succ@0.5pp
EvoTac (on)	**0.4151**	**0.5084**	**0.8846**	**0.70**
w/o M_{self}	0.5324	0.6407	0.8167	0.50
w/o M_{beh}	0.5835	0.7780	0.7297	0.54
w/o M_{strat}	0.6059	0.8248	0.6963	0.56

Table 3: Ablation results removing one memory layer at a time. Each layer contributes to overall performance.

3.3.4 RQ3 Analysis: Impact of Online Self-Evolution

Turning on online rewriting yields consistent improvements over frozen memory: MAE drops from 0.4468 to 0.4151, RMSE from 0.5621 to 0.5084, and Success@0.5pp increases from 0.68 to 0.70. These gains indicate that EvoTac can convert terminal outcomes into targeted heuristic edits that remain beneficial for subsequent instances, rather than requiring periodic retraining.

Empirically, the benefit is most visible in (i) **cold-start entities** with limited history, where online updates quickly calibrate M_{beh} (e.g., acceptance ranges and drift cues), and (ii) **distribution shifts** (e.g., market or platform changes), where layer-wise diagnosis enables selective edits to the affected layer (e.g., M_{self} bounds or M_{beh} drift) while preserving stable cross-context priors in M_{strat}.

3.3.5 Ablation Study: Memory Layers

We further study layer contributions by removing one component at a time (Table 3). All variants degrade, reflecting complementary roles:

- **w/o M_{self}**: Success@0.5pp drops from 0.70 to 0.50, consistent with losing feasibility-aligned anchors and guardrails.

- **w/o M_{beh}**: R^2 falls from 0.8846 to 0.7297, indicating weaker tracking of short-horizon acceptance drift.

- **w/o M_{strat}**: MAE rises from 0.4151 to 0.6059, suggesting that stance/type hypotheses provide important cross-context generalization.

Overall, the "self bounds–behavioral calibration–strategic type" decomposition separates fast drift from slower, reusable structure, supporting targeted updates from terminal feedback and more stable first-offer anchoring.

4 Related Work

Related work on real-world negotiation and games broadly falls into three lines: (i) numerical equilibrium computation and early supervised learning, (ii) multi-agent reinforcement learning, and (iii) LLM-driven self-play and memory-augmented agents. Unlike equilibrium computation or one-shot policy prediction, our focus is on extracting and evolving *reusable, situation-conditioned tacit heuristics* from sparse terminal outcomes.

Numerical equilibrium computation and early ML. Classical work formalizes games and solves (or approximates) equilibria, from Lemke–Howson to bounded-rationality concepts such as QRE (Lemke and Howson, 1964; McKelvey and Palfrey, 1995). These methods are reliable when utilities, strategy spaces, and information structures are well-specified, but real negotiations rarely expose such complete structure. Early ML instead fits mappings from context to outcomes, offering strong baselines under stable distributions but degrading under opponent heterogeneity and mechanism drift; updates typically require offline retraining and provide limited interpretability under terminal-only feedback.

Multi-agent reinforcement learning (MARL). MARL operationalizes equilibrium approximation and best-response learning through self-play. Methods such as PSRO, NFSP, and Deep CFR scale policy-space search and self-play iteration to produce strong approximate equilibria in complex adversarial settings (Lanctot et al., 2017; Heinrich and Silver, 2016; Brown and Sandholm, 2019), enabling milestone systems including AlphaStar and OpenAI Five (Vinyals et al., 2019; Berner et al., 2019). However, real-world negotiation often lacks high-fidelity simulators and dense step-level rewards, providing only terminal outcomes with unobserved trajectories; this makes it hard to run faithful self-play or to continuously adapt from deployment feedback.

LLM-driven self-play and hybrid augmentation. LLMs represent states, histories, and constraints in language space, enabling strategy generation with readable rationales and reducing reliance on explicit structural specification. Systems such as Cicero combine language-based negotiation with strategic reasoning, demonstrating scalable decision-making and explanation in complex games (Bakhtin et al., 2022). Recent hybrid paradigms integrate self-play or preference optimization with tools and explicit memory to incorporate terminal feedback under deployment constraints, including tool-augmented negotiation agents with adaptive opponent modeling (Kwon et al., 2025), self-play-style optimization toward stable strategies (Wu et al., 2024), and compressing failures into reusable heuristics via reflection and memory (Shinn et al., 2023; Packer and Fang, 2023).

A growing line on **LLM self-evolution** systematizes what evolves (policies/tools/memory/data), which feedback drives evolution, and how to evaluate and govern safety (Gao et al., 2025). Collectively, these works suggest that static prompting or retrieval is insufficient under long-term opponent drift and mechanism changes; instead, a closed loop of "terminal feedback $\rightarrow$ attributable diagnosis $\rightarrow$ structured updates" is key for accumulating and correcting tacit game knowledge (Wu et al., 2025; Li et al., 2025).

5 Conclusion

This paper proposes EvoTac, a self-evolving framework that directly targets a central challenge in real-world negotiation: transforming sparse terminal outcomes into reusable tacit negotiation heuristics that generalize under partial observability and opponent drift. Inspired by the human accumulation of social tacit knowledge, EvoTac employs a three-layer decoupled memory mechanism for fine-grained error attribution and continual learning. Experiments in real-world marketing negotiation scenarios show that EvoTac improves final-commission prediction accuracy, producing calibrated first-offer anchoring that improves negotiation efficiency, offering a methodological perspective for developing intelligent agents endowed with social competence and self-evolution capabilities.

Limitations

While EvoTac demonstrates strong performance in extracting tacit rules from terminal outcomes, our evaluation is currently constrained by the difficulty of obtaining high-quality real-world datasets for strategic bargaining settings. EvoTac's predict–diagnose–update loop crucially depends on (i) reliable terminal settlement labels, (ii) sufficiently rich and standardized pre-deal context fields to construct stable scene signatures and actionable guardrails, and (iii) consistent counterparty identifiers to support cross-episode aggregation for opponent behavior and stance hypotheses. In many real negotiation domains, these ingredients are hard to curate at scale: interactions and settlements are often fragmented across channels, recorded inconsistently, and protected by privacy or contractual restrictions, making it non-trivial to release or even internally consolidate a clean outcome-only stream suitable for continual rule evolution under partial observability. As a result, we validate EvoTac in a single domain and downstream task—regressing the final commission rate in influencer-marketing negotiation—and further studies across additional bargaining scenarios and outcome types are needed to assess its generalization.

Ethical Statement

This work focuses on developing EvoTac, an LLM-based self-evolving framework for real-world negotiations, aiming to extract reusable tacit heuristics from terminal outcomes to enhance negotiation efficiency. We fully acknowledge the potential ethical implications of handling negotiation-related data and deploying intelligent negotiation agents, and have taken targeted measures to address them. The dataset used in this study has undergone strict pre-anonymization processing, with all Personally Identifiable Information (PII) of negotiation participants and related entities replaced with generic unique identifiers to eliminate the risk of personal information leakage. Data collection and usage strictly adhere to relevant ethical standards and data governance policies, and the dataset is exclusively used for academic research purposes without any commercial or unauthorized application. To ensure content safety, EvoTac's underlying LLM backbone strictly follows alignment principles to filter out harmful, unethical, or non-compliant content, and sampled instances have been audited by humans to ensure no offensive or toxic content is included.

The framework's predict–reflect–update loop is constrained by ethical guardrails, preventing the generation of risky or malicious content. When deploying EvoTac in practical scenarios, we recommend establishing transparent data usage policies and ensuring informed consent from relevant stakeholders. Future work will continue to prioritize ethical considerations, including optimizing bias mitigation strategies and exploring robust privacy-preserving technologies, to promote the responsible development and deployment of intelligent negotiation technologies.

References

Patrick Bajari, C. Lanier Benkard, and Jonathan Levin. 2007. Estimating dynamic models of imperfect competition. *Econometrica*, 75(5):1331–1370.

Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, Athul Paul Jacob, Mojtaba Komeili, Karthik Konath, Minae Kwon, Adam Lerer, Mike Lewis, Alexander H. Miller, Sasha Mitts, Adithya Renduchintala, Stephen Roller, and 7 others. 2022. Human-level play in the game of Diplomacy by combining language models with strategic reasoning. *Science*, 378(6624):1067–1074.

Christopher Berner, Greg Brockman, and Brooke Chan. 2019. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*.

Noam Brown and Tuomas Sandholm. 2019. Deep counterfactual regret minimization. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97, pages 793–802. PMLR.

Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. 2007. Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *American Economic Review*, 97(1):242–259.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, Hongru Wang, Han Xiao, Yuhang Zhou, Shaokun Zhang, Jiayi Zhang, Jinyu Xiang, Yixiong Fang, Qiwen Zhao, Dongrui Liu, and 6 others. 2025. A survey of self-evolving agents: On path to artificial super intelligence. *Preprint*, arXiv:2507.21046.

Johannes Heinrich and David Silver. 2016. Deep reinforcement learning from self-play in imperfect-information games. In *Advances in Neural Information Processing Systems*, volume 29, pages 53–61. Curran Associates, Inc.

Deuksin Kwon, Jiwon Hae, Emma Clift, Daniel Shamsoddini, Jonathan Gratch, and Gale Lucas. 2025. Astra: A negotiation agent with adaptive and strategic reasoning via tool-integrated action for dynamic offer optimization. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 16217–16238. Association for Computational Linguistics.

Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. 2017. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 30, pages 4190–4203. Curran Associates, Inc.

Carlton E. Lemke and Joseph T. Howson, Jr. 1964. Equilibrium points of bimatrix games. *Journal of the Society for Industrial and Applied Mathematics*, 12(2):413–423.

Dahuang Li, Baolin Peng, Xueying Zhang, and Haizhou Li. 2025. Agentkb: A structured experience repository for continual correction of tacit game knowledge. *Preprint*, arXiv:2507.06229.

Jiaye Lin, Yifu Guo, Yuzhen Han, Sen Hu, Ziyi Ni, Licheng Wang, Mingguang Chen, Hongzhang Liu, Ronghao Chen, Yangfan He, Daxin Jiang, Binxing Jiao, Chen Hu, and Huacan Wang. 2025. SE-Agent: Self-evolution trajectory optimization in multi-step reasoning with LLM-based agents. *Preprint*, arXiv:2508.02085.

Wenxuan Liu, Xiyuan Zhou, Xinlei Wang, Yuheng Cheng, Lixin Ye, Randall Berry, Leandros Tassiulas, Jianwei Huang, and Junhua Zhao. 2025. An LLM agent-based framework for analytical characterization of Nash equilibria. *Nexus*. Commonly referred to as PrimeNash.

Yucheng Luo, Kanghua Yin, Liang Feng, Qian Zheng, Yupeng Hou, Yuanzhi Li, and Min Zhang. 2025. AgentEvolver: Towards efficient self-evolving agent system. *Preprint*, arXiv:2511.10395.

Richard D. McKelvey and Thomas R. Palfrey. 1995. Quantal response equilibria for normal form games. *Games and Economic Behavior*, 10(1):6–38.

Charles Packer and Vivian Fang. 2023. Memgpt: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*.

Jiang Rong, Tao Qin, and Bo An. 2015. Computing quantal response equilibrium for sponsored search auctions. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1803–1804, Istanbul, Turkey. International Foundation for Autonomous Agents and Multiagent Systems.

Noah Shinn, Beck Labash, and Ashwin Gopinath. 2023. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*.

Hal R. Varian. 2007. Position auctions. *International Journal of Industrial Organization*, 25(6):1163–1178.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, and 23 others. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354.

Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, and Botian Shi. 2025. Evolver: Self-evolving LLM agents through an experience-driven lifecycle.

Yue Wu, Zhiqing Sun, Huizhuo Yuan, Kaixuan Ji, Yiming Yang, and Quanquan Gu. 2024. Self-play preference optimization for language model alignment. *arXiv preprint arXiv:2405.00675*.

Kaixuan Xu, Jiajun Chai, Sicheng Li, Yuqian Fu, Yuanheng Zhu, and Dongbin Zhao. 2025. DipLLM: Fine-tuning LLM for strategic decision-making in diplomacy. *arXiv preprint arXiv:2506.09655*. Accepted to ICML 2025.

A Three-Layer Memory Structure Details

The field schema of the three-layer memory is shown in Table 4. We emphasize that the three layers are jointly used for retrieval and prompt construction, while remaining decoupled for writing and rewriting. Each memory unit is stored as "index key (key) + structured fields (value) + statistical and confidence meta-information (meta)," enabling auditable and locally reversible updates. A direct benefit of this decoupling is that, when prediction deviations occur, the system can attribute errors more precisely to "bias in the general response template," "recent opponent behavioral drift," or "mismatch in type/stance hypotheses," making updates more targeted and avoiding the destruction of long-term reusable structures by local noise.

An Example: Commission Negotiation between a Brand and an Influencer Promoter

We use the example of "brand A negotiating a commission rate with promoter B" to illustrate how the three-layer memory is instantiated. Such negotiations can be viewed as a bargaining game with incomplete type information: the brand seeks to control commission and contractual costs while ensuring exposure and fulfillment, whereas the promoter weighs short-term earnings and collaboration terms (e.g., exclusivity, creative asset rights, delivery cycle) under schedule and outside-option constraints. In our setting, the observable input for a single instance is the profiles and context $x_t = \langle \rho_t^{(A)}, \rho_t^{(B)}, \mathcal{C}_t \rangle$, and the observable feedback is only the terminal settlement y_t (e.g., the final commission in percentage points), without intermediate offer trajectories.

Table 5 provides a field-level illustration of the three memory layers for a single matchmaking instance. Intuitively, $z = g(\mathcal{C}_t)$ corresponds to "what is the current situation/constraints": it abstracts key dimensions directly relevant to equilibrium response, used to retrieve our response boundaries and leverage-exchange templates under the situation. The opponent signature $o = h(\rho_t^{(B)})$ corresponds to "who the opponent is / what class of opponent it belongs to": it maps opponent profiles to a stable index, enabling cross-instance aggregation for the same or similar promoters. Based on aggregated estimates over the historical set $\{(x, y)\}$, M_{self} provides our stable objectives/constraints and equilibrium anchor under z (and "which clauses can substitute for price concessions"); M_{beh} provides promoter B's recent acceptance interval, clause sensitivity, and drift under similar sub-scenarios; M_{strat} provides slower type/stance hypotheses and more stable cross-scenario response structures (e.g., how demand rigidity changes with deadline pressure), and specifies the evidence thresholds under which these hypotheses should be updated. During online reasoning, all three sources enter the prompt jointly, enabling the frozen LLM to output numeric anchors as well as interpretable and actionable strategic recommendations.

B Data Privacy and Ethics Statement

To ensure compliance with ethical standards regarding data usage:

- **Anonymization and PII protection:** The dataset provided by the commercial platform was pre-anonymized. All Personally Identifiable Information (PII) covering brands and influencers was replaced with generic unique identifiers (e.g., `SPORTS-CREATOR-02` as shown in Table 5) prior to our access.
- **Content safety:** The dataset strictly covers professional commercial negotiations. A human audit was conducted on the sampled instances to verify the absence of offensive, toxic, or harmful content.

C Implementation Details

This appendix lists the main prompts that EvoTac sends to the backbone LLM, organized by the online workflow: *(1) retrieve memory → (2) predict → (3) diagnose with terminal feedback → (4) write/update memory*. For readability, we present prompt templates with placeholders, and summarize the required output schema for each step. Unless otherwise stated, we use a fixed top-K retrieval per memory layer and keep this retrieval setting consistent across all experiments.

C.1 Parameter Setup

We use the following hyper-parameters for all experiments to ensure reproducibility:

- **Retrieval configuration:** We use an embedding encoder for vector retrieval over each memory layer. The encoder is fixed throughout a single experiment run.
  - Encoder models: DashScope `text-embedding-v1`.

Memory layer	Game-theoretic role	Index key (Key)	Core fields (Value + Meta)
M_{self} Self-strategy memory	Our stable objectives/constraints and equilibrium response structure under a given situation (reusable policy template)	Scenario signature z: market/category, price tier, opponent type, supply/demand, seasonality, platform constraints	Situation signature: market/category, price tier, opponent type, constraints; Preconditions: brand tier and campaign goals (only the determinants of the feasible region); Strategy template: anchor / first-offer rule, concession schedule, non-price levers (clauses) and triggers, messaging frame; Guardrails: min/max commission, ROI floor, risk limits; Evidence: supporting historical records (IDs, outcomes, tags); Meta: stability, drift, sample size, success rate, confidence, update history, timestamps
M_{beh} Opponent-behavior memory	Empirical beliefs about the opponent in the observable outcome space (sensitive to short-term non-stationarity/drift)	Opponent signature o: affiliate ID (unique identifier for cross-instance aggregation)	Opponent ID: unique identifier for the opponent; Outcome statistics: recent accept range, mean/std (or quantiles), accept rate (and sub-scenario conditioning if needed); Drift cues: short-term shifts (e.g., upward/downward move), response-speed/flexibility indicators, clause sensitivity; Evidence: time-stamped (context, tag, outcome) traces; Meta: stability, drift, fit error, interaction counts, confidence, update history, timestamps
M_{strat} Opponent-strategy memory	Slow-varying hypotheses about opponent type/stance and more stable cross-scenario response structure (requires strong evidence accumulation)	Opponent signature o: affiliate ID + brand profile context	Opponent ID + context key: affiliate ID and coarse brand/profile context; Stance hypothesis: stance label and supporting cues; Stable response structure: anchor aggressiveness, concession speed, deadline/constraint sensitivity, switching conditions; Reservation/exit: acceptable range and termination/hardline triggers; Guardrails: sweet spot / minimum acceptable, risk watch points; Evidence: cross-scene records; Meta: stability, drift, support size, confidence, update history, timestamps

Table 4: Structured fields of EvoTac's three-layer decoupled memory.

  - Vector index: FAISS `IndexFlatL2` for $M_{\text{self}}, M_{\text{beh}}, M_{\text{strat}}$.
  - Ranking metric: L2 distance d (convenience similarity $s = 1/(1+d)$ is used for logging).
- **Memory retrieval bounds (Top-K):**
  - Self-strategy memory (K_{self}): 5
  - Opponent-behavior memory (K_{beh}): 3
  - Opponent-strategy memory (K_{strat}): 3
- **Alignment and reflection:** Terminal-only labeling utilizes the scenario-normalized error $\varepsilon_t = \frac{|\hat{y}_t - y_t|}{u_t - l_t + \eta}$, where $[l_t, u_t]$ is the brand-provided feasible range and $\eta = 0.1$.
  - Aligned threshold (δ_1): 0.15 (Aligned if $\varepsilon_t \le \delta_1$).
  - Misaligned threshold (δ_2): 0.25 (Misaligned if $\varepsilon_t \ge \delta_2$).
  - *Note*: Values between δ_1 and δ_2 are treated as neutral to suppress over-updating.
- **Cost and latency budgets:**
  - Predictor: `max_tokens`=16000, timeout=180s (up to 3 retries).
  - Experience reflector: `max_tokens`=2000.
  - Memory manager: `max_tokens`=8000.
  - Embedding: Batch size 5 (1s inter-batch delay).

C.2 Prompt Engineering Details

We provide instance-level prompt templates for the three prompt-driven LLM calls in the predict-reflect-update loop: prediction, terminal-feedback diagnosis, and memory writing/rewriting.

C.2.1 Commission Prediction (EvoTac Predictor)

Purpose. Given a brand profile, an affiliate profile, and the retrieved three-layer memories (vector retrieval), the predictor outputs a single commission estimate with rationale and cited memory IDs.

Template (placeholders shown in curly braces {}).

```
[SYSTEM]
You are a commission prediction agent under the
EvoTac framework. You must predict an appropriate
commission rate based on a three-layer decoupled
memory.

EvoTac three-layer memory architecture:
1. Self-strategy memory: situation--policy pairs
that provide reusable context and strategy
templates.
2. Opponent-behavior memory: observable opponent
behavior patterns, including acceptance ranges
and response patterns.
3. Opponent-strategy memory: opponent stance/type
hypotheses and constraints, including strategy
portrait, stance, and reservation range.
```

Memory layer	Example Entry (Key → Value)
M_{self}	Situation signature: {market category=beauty; price tier=mid-to-high; opponent type=beauty expert; supply/demand=normal; seasonality=[normal]; platform constraints={}} Strategy template: {anchor: "first_offer_pp = hist_accept_p50_pp + 0.3"; concession: [-0.5, -0.3, -0.2]pp/round; acceptable range: [15.5%, 18.0%]; levers: [delivery_cycle, exclusivity, creative_assets]; frames: [scarcity, brand_fit]} Mechanism (optional): {Rubinstein / Nash; discount factors: {0.93, 0.95}} Guardrails: {min_pp: 9.9%, max_pp: 24.75%, roi_floor: 1.5} Metrics: {stability score: 0.75, drift score: 0.0, sample size: 15, success rate: 0.80}
M_{beh}	Affiliate ID: SPORTS-CREATOR-02 Outcome statistics: {accept range: [5.0%, 9.0%], mean: 7.1%, std: 1.5%, accept rate: 0.75} Drift/style cues: {drift score: 0.05, flexibility: moderate} Recent evidence: [{2024-12-30, context: {industry: sports, price position: mid}, tag: SUCCESS_ACCEPT, outcome: 7.099%}, ...] Metrics: {stability score: 0.68, model fit error: 0.40, total interactions: 8 (6/2 accept/reject)}
M_{strat}	Affiliate ID: SPORTS-CREATOR-02; Brand profile: {industry: sports, price position: mid} Type/stance: {stance: cashflow-driven}; stable params: {first offer: moderate, concession speed: moderate, duration: short} Reservation/exit: {acceptable range: [4.0%, 7.5%]; exit/hardline triggers: [offer $<$ 0.85$\times$ hist_success_mean, concession too slow]} Acceptance guardrails: {min_pp: 4.0%, sweet_spot: 6.25%} Metrics: {stability score: 0.62, drift score: 0.0, success/failure: 6/2, acceptance rate: 0.75}

Table 5: Field-level examples of three-layer memory in the commission matchmaking scenario.

```
Prediction requirements:
- Carefully analyze the brand profile, the
affiliate profile, and the retrieved historical
memories (via vector retrieval).
- Jointly consider the situational match of
Self-strategy memory, the historical acceptance
patterns in Opponent-behavior memory, and the
stance constraints in Opponent-strategy memory.
- Produce an explicit reasoning trace and
explicitly cite the memory entries used
(memory_id).
- Explain how the three layers are combined to
reach the final prediction.
- Provide a reasonable commission prediction and
a confidence score.

Output format requirements:
{format_instructions}

[HUMAN]
### Brand profile
{brand_block}

### Affiliate profile
{affiliate_block}

### Retrieved memories (vector retrieval;
three-layer decoupled memory)
{memory_block}

Based on the above information, make a prediction
under EvoTac:
1. Self-strategy memory: which strategy
templates match the current context (industry,
price positioning, affiliate category)?
2. Opponent-behavior memory: what is this
affiliate's historical acceptance pattern and
typical acceptable commission range?
3. Opponent-strategy memory: what is this
affiliate's stance and bottom line, and what
negotiation style might they adopt?
4. Combine the three layers and provide the final
prediction with reasoning.
Please explicitly cite the specific memory IDs
(memory_id) and justify the prediction.
```

Output schema (fields).

- `predicted_commission_pp`: predicted commission (percentage points).
- `reasoning`: textual rationale citing memories.
- `memory_refs`: referenced `memory_ids`.
- `confidence`: confidence score in $[0, 1]$.

C.2.2 Terminal-Feedback Diagnosis (Experience Reflector)

Purpose. After the terminal settlement is observed, the Experience Reflector assigns an alignment label and produces actionable feedback (including memory update suggestions) from terminal-only supervision.

Note. The prompt always includes a "negotiation trajectory" block (`negotiation_block`); for the single-agent predictor used in our commission prediction setting, this block is empty and explicitly marked as such by the formatter in code.

Template (placeholders shown in curly braces {}).

```
[SYSTEM]
You are an Experience Reflector
(distortion-attribution analyzer) under the
EvoTac framework. You must analyze a predictor's
result,
determine whether it is aligned, misaligned, or
neutral, identify the key responsible party(ies),
and provide actionable improvement suggestions.
```

```
Please follow the EvoTac Experience-reflector
framing.

Decision criteria:
- aligned: prediction error is small (typically
gap_pp <= 0.4 or relative error <= 8%).
- misaligned: prediction error is large
(typically gap_pp > 0.5 or relative error > 10%)
and requires attribution.
- neutral: error is between aligned and
misaligned and may need mild adjustment.

EvoTac three-layer memory architecture:
1. Self-strategy memory: situation--policy pairs
(reusable context).
2. Opponent-behavior memory: observable opponent
behavior patterns.
3. Opponent-strategy memory: opponent
stance/type hypotheses and constraints.

Responsibility vector (responsibility_vector):
Assign responsibility weights across the
following components (summing to 1.0):
- environment_encoding: context encoding error
(e.g., misunderstanding market category or price
tier).
- memory_retrieval: retrieval error (e.g.,
retrieving irrelevant memories).
- strategy_planning: planning error (e.g.,
selecting an inappropriate strategy template).
- response_generation: generation error (e.g.,
LLM output deviation).
- exogenous_factors: exogenous factors (e.g.,
market shifts, opponent temporary changes).

Error type tags (error_type_tags):
Common error types include:
- 'inappropriate situational strategy': chosen
strategy is unsuitable for the current context.
- 'biased opponent-behavior estimate': the
opponent behavior pattern is inaccurately
estimated.
- 'incorrect opponent-stance inference': the
opponent stance/bottom line is incorrectly
inferred.
- 'ignored platform-level constraints': platform
constraints were ignored.
- 'inaccurate retrieval': retrieved memories do
not match the current context.
- 'poor strategy parameterization': parameters
(e.g., first offer, concession pace) are poorly
set.

Memory update suggestions
(memory_update_suggestions):
Provide suggestions for each memory layer:
- self_strategy_memory: suggested operation
(create/update/merge/deprecate) and what to
change.
- opponent_behavior_memory: suggested operation
and what to change.
- opponent_strategy_memory: suggested operation
and what to change.

Notes:
- If a strategy-planning output is provided
(plans field), analyze whether the selected
memories are appropriate.
- If aligned, responsible_agents should be an
empty list or include 'none', and
root_causes/actionable_feedback should be empty.
- If retrieved memories are provided
(retrieved_memories), analyze whether they match
the current context.
Output JSON Schema:
{
  "alignment_status": "aligned | misaligned |
  neutral",
  "responsible_agents": [
    "platform | influencer | both | none"
  ],
  "root_causes": [
    "string"
  ],
  "actionable_feedback": {
    "platform": "string (optional)",
    "influencer": "string (optional)",
    "both": "string (optional)"
  },
  "confidence": "number in [0, 1]",
  "summary": "string",
  "responsibility_vector": {
    "environment_encoding": "number",
    "memory_retrieval": "number",
    "strategy_planning": "number",
    "response_generation": "number",
    "exogenous_factors": "number"
  },
  "error_type_tags": [
    "string"
  ],
  "memory_update_suggestions": {
    "self_strategy_memory": "object",
    "opponent_behavior_memory": "object",
    "opponent_strategy_memory": "object"
  }
}

[HUMAN]
### Real-world feedback
{real_outcome_block}

### Model alignment metrics
{alignment_metrics_block}

{plans_block}{retrieved_memories_block}
Please complete the diagnosis based on the above
information, focusing on:
1. Determine alignment status
(aligned/misaligned/neutral).
2. Analyze the responsibility vector and identify
the main faulty component.
3. Assign error type tags.
4. Provide concrete update suggestions for the
three memory layers
(create/update/merge/deprecate).
```

Output schema (high level). The Experience reflector returns a JSON object with (a) an alignment status in {aligned, neutral, misaligned}; (b) a responsibility vector over environment encoding, memory retrieval, strategy planning, response generation, and exogenous factors; and (c) memory update suggestions for self-strategy, opponent-behavior, and opponent-strategy memory.

C.2.3 Memory Writing / Rewriting (Memory Manager)

Purpose. Convert each episode (success or failure) into structured, reusable memory entries. The implementation enforces a strict "JSON-only" constraint and provides an explicit Schema in the prompt.

Invocation wrapper. For each memory layer, the system sends a short system instruction that enforces *JSON-only* output, followed by a composed user prompt. The placeholder `{composed_prompt}` denotes this fully assembled prompt text after filling in the placeholders in the templates below (e.g., inserting the episode-specific `{context_json}` and the corresponding schema/rules). We show the wrapper and the layer-specific templates separately for clarity.

```
[SYSTEM]
You are a rigorous data analysis assistant.
Output only JSON that satisfies the requirements.

[HUMAN]
{composed_prompt}
```

Prompt template (placeholders shown in curly braces {}).

(a) Self-strategy memory.

```
You are a memory distillation expert for a
matchmaking platform. Your job is to distill
successful and failed deal experiences into
reusable strategy snippets. Focus on transferable
conditions, strategies, guardrails, and signals.

Read the structured data below and output strict
JSON:
Context:
{context_json}

Output rules:
1. Output JSON only; no extra text or comments.
2. The JSON must satisfy the following schema:
{
  "title": "string",
  "description": "string",
  "policy": {
    "first_offer_rule": "string",
    "concession_schedule_pp": "number[]",
    "messaging_frames": "string[]",
    "disclosure_assets": "string[]"
  },
  "mechanism": "object",
  "behavioral_cues": "array",
  "guardrails": "object"
}
3. You may add explanatory text, but do not
fabricate numbers absent from the input.
4. If the input contains a mode field,
distinguish the tone of create/update.
- In the description, extract key metrics (e.g.,
success rate, sample size, concession strategy).
- policy.concession_schedule_pp must strictly
follow the numeric order provided in the input.
- Numeric values in guardrails must match the
input (you may add explanatory fields).
```

(b) Opponent-behavior memory.

```
You are an opponent behavior analyst. Summarize
the opponent's negotiation style, acceptance
boundaries, and actionable insights into a
queryable memory entry.

Read the structured data below and output strict
JSON:
Context:
{context_json}

Output rules:
1. Output JSON only; no extra text or comments.
2. The JSON must satisfy the following schema:
{
  "title": "string",
  "description": "string",
  "behavior_pattern": "object",
  "negotiation_style": "object"
}
3. You may add explanatory text, but do not
fabricate numbers absent from the input.
4. If the input contains a mode field,
distinguish the tone of create/update.
- Numeric ranges/ratios in behavior_pattern must
match the input.
- negotiation_style may include extra
explanation, but must remain logically
consistent.
```

(c) Opponent-strategy memory.

```
You are an affiliate strategy analyst. Summarize
negotiation stance and acceptance guardrails.
Extract executable strategy insights for
downstream agent use.

Read the structured data below and output strict
JSON:
Context:
{context_json}

Output rules:
1. Output JSON only; no extra text or comments.
2. The JSON must satisfy the following schema:
{
  "title": "string",
  "description": "string",
  "stance": "string",
  "bargaining_tactics": {
    "acceptance_guardrails": "object",
    "preferred_counter_patterns": "array",
    "risk_profile": "string"
  },
  "guardrails": "object"
}
3. You may add explanatory text, but do not
fabricate numbers absent from the input.
4. If the input contains a mode field,
distinguish the tone of create/update.
- Focus on the affiliate-side payoff
requirements, risk preferences, and common
concession patterns.
- Use guardrails to describe non-negotiable
constraints or watch-point triggers.
```

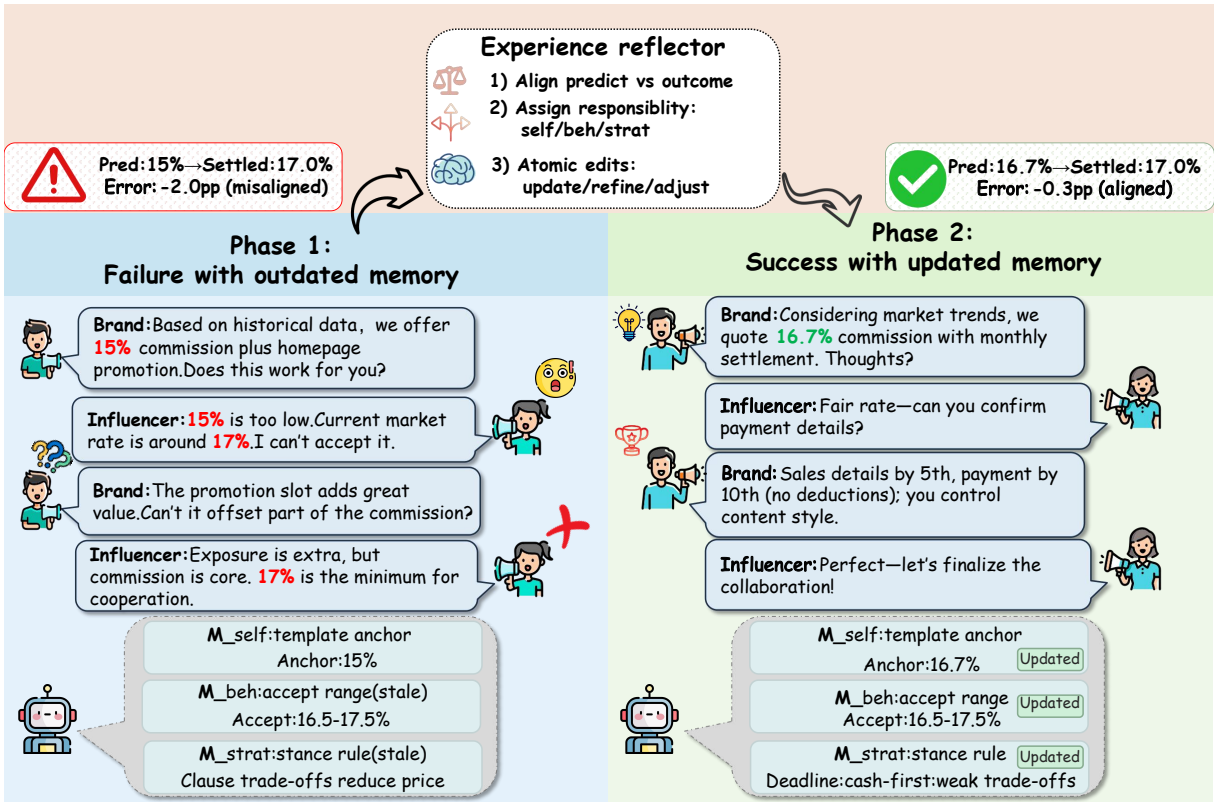

Figure 3: Case study: self-evolving memory for negotiation pricing.

D Case Study

EvoTac is deployed in a large internet marketing platform's *brokered* commission negotiation setting: the platform matches a brand with a promoter and provides profiles, historical statistics, and a recommended commission band, while EvoTac only observes the **final settled commission**. Since the interaction trajectory is unobserved, EvoTac is used as a *terminal-outcome predictor*: its output serves as a calibrated first-offer anchor to reduce rounds. Figure 3 illustrates two comparable episodes in the same sub-scenario, highlighting how three-layer memory supports a *predict–reflect–update* loop under terminal-only supervision.

Phase 1: Failure with outdated memory. In the first match, the brand opens too low: *"we offer 15% commission plus homepage promotion."* The promoter rejects and sets a hard floor: *"commission is core. 17% is the minimum."* The deal settles at 17.0%, yielding a clear underestimation (Pred: 15.0% → Settled: 17.0%, error −2.0pp). The root cause is miscalibrated retrieval: M_{self} anchors on an outdated 15% template; M_{beh} reflects an older acceptance interval (e.g., 15.5–16.5%) and misses the upward drift; and M_{strat} overestimates the effectiveness of non-price trade-offs, delaying the necessary upward move on commission.

Evolution core: Attributing the deviation and updating the right layer. After observing only the terminal settlement, the Experience Reflector assigns responsibility across `self/beh/strat`, and the Memory Manager applies minimal, layer-specific edits: update M_{beh} to a refreshed acceptance interval with drift cue (e.g., 16.9–17.5% and rising), refine M_{strat} into a constraint-sensitive stance heuristic (e.g., under tight deadlines, *cash-first* and weak trade-offs), and adjust M_{self} to a higher *neutral* anchor closer to feasibility (e.g., 16.8%) rather than tying the anchor to the final quote.

Phase 2: Success with updated memory. In a subsequent similar match, retrieval surfaces the updated M_{beh} and refined M_{strat}, so the brand opens near the feasible region: *"we quote 16.9% commission with monthly settlement."* The promoter makes a small upward request to the floor and moves to execution (*"17.0% is the minimum—confirm payment details?"*), and the brand accepts while confirming key terms (e.g., payment schedule and no-deduction policy). The deal settles at 17.0% with a small error (Pred: 16.9% → Settled: 17.0%, er-

ror -0.1pp), improving one-shot calibration and reducing rounds.

Overall, the case study shows why three-layer separation matters under terminal-only feedback: M_{self} provides reusable feasibility-aligned anchors, M_{beh} tracks fast drift in acceptance dynamics, and M_{strat} captures slower stance/constraint response, enabling targeted updates instead of noisy global rewrites.