Unifying Multi-Role Agents into a Single LLM for Controllable Multi-Path Relevance Annotation

Anonymous ACL submission

Abstract

Relevance annotation is a complex task that requires integrating multiple sources of evidence, such as semantic matching between user queries and item fields, click-intent inference, relevance judgment, and auditing reflection. Multi-agent collaboration is naturally suited for such multi-step reasoning, but conventional methods typically distribute these capabilities across external workflows rather than unifying them within a single model. To address this issue, we propose **URAFM** (**Unified Reasoning Agents Framework Model**), a unified reasoning framework that enables a single model to perform **controllable multi-path reasoning** over a fixed set of reasoning paths. Specifically, we use external multi-agent teachers to synthesize structured supervision data with explicit path labels, and construct a path-balanced training set of 150K samples covering 10 predefined valid paths. We then perform unified supervised fine-tuning so that the model jointly learns path selection, role execution, and final label prediction within a shared parameter space. Building on this capability, we further propose **multi-path expert voting**, which treats different reasoning paths as complementary path experts and aggregates their predictions at inference time to mitigate single-path bias. Experiments on a multilingual food-delivery relevance annotation dataset show that URAFM achieves 0.8170 Accuracy and 0.8183 Weighted-F1, outperforming 8B label-only fine-tuning by 3.47 and 3.30 percentage points, respectively, while reducing total token consumption by 76.2% compared with the conventional multi-LLM workflow. With multi-path expert voting, performance further improves to 0.8399 Accuracy and 0.8414 Weighted-F1. These results demonstrate that unifying multi-role reasoning within a single model and exploiting structured path diversity at inference time are complementary and parameter-efficient sources of gain for relevance annotation.

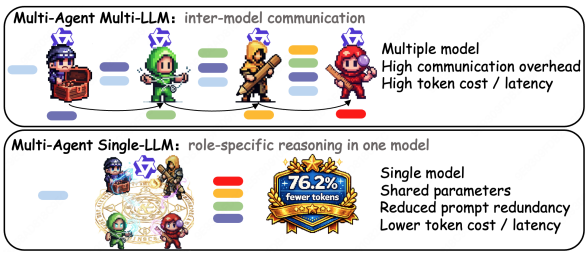

Figure 1: Comparison between the conventional Multi-Agent Multi-LLM paradigm and our Multi-Agent Single-LLM paradigm. The former distributes different roles across multiple external LLMs and relies on cross-model communication for collaboration, leading to substantial prompt redundancy and interaction overhead. The latter unifies multi-role capabilities within a single model and enables controllable multi-path reasoning in a shared parameter space, thereby significantly improving efficiency and reducing inference cost.

1 Introduction

In recent years, large language models (LLMs) have shown strong capability in food-delivery relevance annotation (Ye et al., 2025a; Zhao et al., 2025; Tang et al., 2025; Liu et al., 2024; Mehrdad et al., 2024). However, most existing methods still follow a one-shot reasoning paradigm, i.e., directly predicting the final label or explanation from a query–item pair (Dong et al., 2025; Tang et al., 2025; Mehrdad et al., 2024). While such methods usually work well on simple cases with clear semantics and sufficient evidence, they are often less robust on more difficult cases, such as tail queries, ambiguous intents, or evidence-sparse samples, where multiple types of evidence must be jointly considered for accurate annotation.

In fact, relevance annotation is better viewed as a structured decision-making process: annotators typically analyze semantic matching, infer click-intent, apply annotation rules, and audit the result. Especially for difficult cases, such a step-by-step process is more robust than one-shot reasoning, as

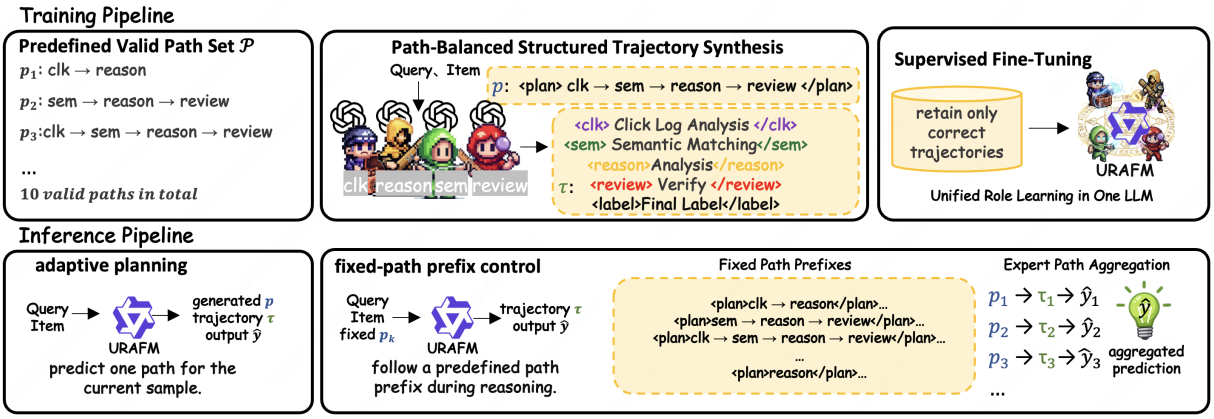

Figure 2: Overview of URAFM. During training, we first define a set of predefined valid paths $\mathcal{P}$ and use external multi-agent teachers to synthesize path-balanced structured supervision data, where the path planning result is denoted by p and the subsequent multi-role structured outputs are denoted by the trajectory τ. We retain only trajectories whose final prediction is correct, yielding a total of 150K training instances, which are then used to fine-tune a single URAFM model to unify multi-role reasoning abilities within a shared parameter space. At inference time, URAFM supports two modes: adaptive planning dynamically predicts one path for each sample and generates the corresponding trajectory, while fixed-path prefix control performs reasoning along a specified predefined path prefix. Based on the latter, we further generate trajectories and predictions along multiple fixed paths and aggregate the outputs of different path experts to obtain a more robust final prediction.

it decomposes heterogeneous evidence into separate stages and reduces local bias. Multi-agent methods naturally fit this paradigm by assigning specialized roles to each stage (Zhu et al., 2025; Hu et al., 2025; Li et al., 2023; Tran et al., 2025), producing more interpretable trajectories and better integrating multiple evidence sources.

However, conventional multi-LLM multi-agent frameworks typically implement different agents as separate models or external workflows, which leads to two key limitations. First, roles must frequently exchange prompts, contexts, and intermediate results, leading to substantial communication and inference overhead. Second, role-specific knowledge and capabilities are fragmented across different modules, making it difficult to achieve effective sharing and transfer within a unified parameter space, which in turn limits cross-role knowledge reuse and overall performance improvement (Zhu et al., 2025; Li et al., 2025a). This raises a key question: how can we preserve the benefits of multi-role structured reasoning while reducing system complexity and improving knowledge integration?

To address this issue, we propose a unified reasoning framework, **URAFM**. Our goal is not merely to reduce the number of models, but to distill the multi-role capabilities originally distributed across external multi-agent collaboration into a single local LLM, so that it can jointly perform path selection, role execution, and label prediction within a shared parameter space. Specifically, we first use external multi-agent teachers to generate structured supervision data with explicit role assignments and path labels, and construct a path-balanced training corpus over 10 predefined valid reasoning paths. We then perform unified supervised fine-tuning so that the model learns to execute multi-role collaboration within a single model, while preserving role boundaries through path prefixes and structured tags. As a result, URAFM can perform **controllable path reasoning** over a fixed path set: it first selects a candidate path and then executes the corresponding roles step by step in a predefined order.

More importantly, this controllable path reasoning capability gives rise to path-level specialization. Because different paths differ in the evidence they emphasize, their step composition, and their decision focus, the same model exhibits distinct conditional prediction behaviors under different path constraints. In other words, URAFM learns not a single decision pattern, but a set of **path experts** that share parameters while specializing through different paths. Based on this observation, we further propose a **multi-path expert voting** mechanism: for each input sample, the model reasons along multiple predefined paths, and the predictions from different paths are aggregated as complementary expert judgments (Gan and Isola, 2026; Wang et al., 2022; Li et al., 2025b). This mechanism effectively exploits path complementarity,

mitigates single-path bias, and further improves the robustness of the final prediction.

Experiments on a multilingual food-delivery relevance annotation task validate the effectiveness of the proposed method. URAFM achieves 0.8170 Accuracy and 0.8183 Weighted-F1 on the test set, outperforming the 8B label-only fine-tuning baseline by 3.47 and 3.30 percentage points, respectively, and also surpassing the conventional multi-LLM multi-agent workflow. With multi-path expert voting, performance further improves to 0.8399 Accuracy and 0.8414 Weighted-F1. Meanwhile, the total token consumption is only about one quarter of that of the traditional workflow, showing that the proposed method substantially reduces inference cost while improving performance.

The main contributions of this work are as follows:

- We propose **URAFM**, a unified reasoning framework that distills multi-role structured reasoning capabilities from external multi-agent workflows into a single large language model, enabling controllable reasoning over a fixed set of valid paths.
- We introduce a **multi-path expert voting** mechanism that treats different reasoning paths as complementary experts and aggregates their predictions at inference time, thereby reducing single-path bias and improving robustness and accuracy.
- We conduct extensive experiments on a multilingual food-delivery relevance annotation task. The results show that URAFM outperforms single-path baselines and multi-LLM multi-agent workflows while reducing total token cost by **76.2%**. Furthermore, aggregating more expert paths consistently yields additional performance gains.

2 Related Work

2.1 Relevance Modeling

Relevance modeling estimates semantic alignment between a query and a candidate item. Early methods used hand-crafted features (BM25, TF-IDF); later, BERT-style models learned dense representations. More recently, LLMs have advanced relevance modeling through data synthesis (Tang et al., 2025), data filtering (Dong et al., 2025; Liu et al., 2024), model training (Liu et al., 2024; Gan et al., 2025b; Ye et al., 2025a), and distillation (Zhao et al., 2025). Despite these advances, most methods rely on a single LLM backbone subject to hallucination and unstable reasoning. In contrast, we formulate food-delivery relevance annotation as structured multi-step reasoning, unifying role-specialized steps within a single model via path-supervised learning.

2.2 Multi-Agent LLMs

Multi-Agent LLMs have recently emerged as a promising paradigm for solving complex tasks (Gan et al., 2025a; Ye et al., 2025b; Yuksel et al., 2025). The key idea is to decompose a task into multiple specialized agents that collaborate through role division, communication, and coordinated decision-making. Early works such as CAMEL and AutoGen explored role-playing and multi-turn interaction as basic collaboration paradigms, showing that multi-agent communication can improve problem decomposition and task solving (Li et al., 2025a, 2023; Wu et al., 2024). Subsequent studies further extended this line of research from the perspectives of organizational structure and collaboration mechanism. For example, AgentVerse formalizes multi-agent collaboration into multiple stages, MetaGPT improves efficiency through structured outputs, and OWL highlights the importance of multi-role workflow design for real-world task automation (Hu et al., 2025; Chen et al., 2023).

Recent work has begun to systematically analyze the multi-agent design space, finding that effectiveness depends on role assignment, communication structure, and orchestration strategy (Tran et al., 2025; Zhu et al., 2025). A closely related paradigm is Chain-of-Agents (Li et al., 2025a), which focuses on end-to-end multi-agent collaboration distillation, primarily concerned with task decomposition and inter-agent information passing efficiency. Unlike these approaches, URAFM treats **path controllability as a first-class design principle**: path-balanced supervision during training explicitly preserves the heterogeneity of all 10 predefined paths, and fixed-path prefix control at inference time instantiates the same model as multiple complementary path experts. This **path-level controllability and structured diversity exploitation**—enabling deterministic multi-path expert voting via a single model—is the key distinction from Chain-of-Agents and general agent-workflow distillation methods.

Unlike conventional external multi-agent sys-

Non-Agent Baselines					
Model	**Acc**	**F1**	**Acc-0**	**Acc-1**	**Acc-2**
BGE-M3 (Label-only Fine-tuning)	0.7792	0.7882	0.7803	0.7774	0.7800
mBERT (Label-only Fine-tuning)	0.7760	0.7759	0.8438	0.6894	0.7947
Qwen3-8B (Zero-shot)	0.7529	0.7688	0.7782	0.6176	0.8629
Qwen3-8B (Label-only SFT)	0.7823	0.7853	0.8353	0.7106	0.8009
Multi-Agent Models					
Qwen3-8B + External Multi-LLM Workflow	0.7926	0.8041	0.8400	0.8079	0.7300
Teacher LLM + External Multi-LLM Workflow	0.8014	0.8024	0.8503	0.7641	0.7897
URAFM (Weak Teacher / Self-Teacher)	0.7912	0.7936	0.7859	0.7859	0.8018
URAFM (Adaptive Planning)	0.8170†	0.8183	0.8541	0.8168	0.7797
URAFM + Multi-Path Expert Voting ($N = 5$)	0.8319†	0.8334	0.8508	0.8376	0.8071
URAFM + Multi-Path Expert Voting ($N = 9$)	**0.8399**†	**0.8414**	**0.8565**	**0.8590**	**0.8041**

Table 1: Performance comparison on the multilingual food-delivery relevance annotation task. Non-agent baselines include zero-shot LLMs, label-only SFT, and embedding-based models. Multi-agent models include external multi-LLM workflows and URAFM variants. **Teacher LLM** uses Gemini 2.5 Pro.**Weak Teacher** uses untuned Qwen3-8B as the teacher instead of Gemini 2.5 Pro. † denotes statistically significant improvement over **Qwen3-8B (Label-only SFT)** by McNemar's test ($p < 0.001$, $N = 10{,}000$). URAFM (Adaptive Planning) vs. Qwen3-8B Label-only SFT: $+3.47\%$, $\chi^2 = 41.94$, $p < 0.0001$; vs. Teacher LLM + External Workflow: $+2.44\%$, $\chi^2 = 37.54$, $p < 0.0001$.

tems, our method realizes multi-agent reasoning within a single LLM, thereby reducing communication overhead and capability fragmentation.

3 Method

The overall framework is illustrated in Figure 2. We first use a stronger external LLM-driven multi-agent teacher to synthesize path-balanced multi-role structured supervision data offline along predefined valid reasoning paths. We then perform unified supervised fine-tuning on the base model, enabling it to learn path planning, role execution, and label prediction within a single parameter space. Finally, at inference time, we perform multi-path reasoning through fixed-path prefix control and obtain the final prediction via multi-path expert voting.

3.1 Multi-Role Structured Supervision for URAFM

As shown in Figure 3, the external multi-agent teacher consists of four core roles: the **Click-Intent Agent** (`clk`), the **Semantic Matching Agent** (`sem`), the **Annotation Agent** (`reason`), and the **Auditing Reflection Agent** (`review`). Specifically, `clk` retrieves high-CTR query neighbors from click logs and infers the user's latent intent; `sem` analyzes the attribute-level matching relationship between the query and the item; `reason` produces the final relevance judgment by integrating multiple sources of evidence; and `review` audits the previous reasoning and corrects potential errors. These roles are not used as the final deployment paradigm. Instead, we use them as a **structured data construction mechanism** to decompose complex relevance annotation into multi-step reasoning trajectories with explicit step boundaries.

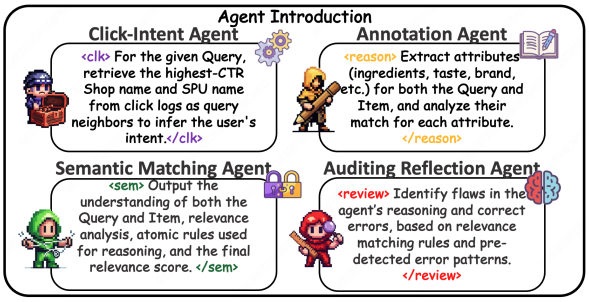

Figure 3: The four core roles of the external multi-agent teacher: click-intent inference, semantic matching, relevance annotation, and auditing reflection. Together, they form the multi-role structured annotation pipeline used for offline data synthesis.

Specifically, we predefine a fixed set of valid paths

$$\mathcal{P} = \{p_1, p_2, \ldots, p_{10}\},$$

where each path consists of a sequence of role-

specific steps, for example,

$$\texttt{sem} \rightarrow \texttt{clk} \rightarrow \texttt{reason} \rightarrow \texttt{review}.$$

Given a training sample with ground-truth label, $x = (q, i, y)$, we first sample a path p from $\mathcal{P}$ using a path-balancing strategy, and then invoke a stronger external LLM-driven multi-agent teacher under the constraint of that path to generate a candidate reasoning trajectory. We retain only those trajectories whose final prediction matches the ground truth as supervision data. Each training sample is thus represented as

$$\tau = \langle\texttt{plan}\rangle p \langle\texttt{/plan}\rangle \oplus \langle\texttt{trajectory}\rangle, \quad p \in \mathcal{P},$$

where $\langle\texttt{trajectory}\rangle$ contains the intermediate outputs of all roles along the path as well as the final label. This design explicitly preserves both path order and role boundaries, allowing the model to learn **path-conditioned structured reasoning** rather than merely fitting the final label.

During data construction, we enumerate 10 valid paths and synthesize approximately 15K high-quality structured trajectories for each path, resulting in a path-balanced training corpus of about 150K instances in total. At the data level, this design explicitly preserves path heterogeneity, which provides the foundation for subsequent unified learning and multi-path aggregation.

3.2 Unified Reasoning Agents Framework Model (URAFM)

After obtaining the multi-role structured supervision data, we train the Unified Reasoning Agents Framework (URAFM). The key idea is not to simply chain multiple roles together, but to let a **single model jointly learn multiple roles**, thereby consolidating the multi-role reasoning capabilities previously scattered across the external workflow into a unified parameter space. In this way, the model not only learns the input-output patterns of different roles, but also benefits from cross-role knowledge sharing and joint optimization.

We fine-tune the base model on the structured training corpus using the following supervised objective:

$$\mathcal{L}_{\text{SFT}} = -\sum_{t \in \mathcal{O}} \log \pi_\theta(\tau_t \mid \tau_{<t}, x), \tag{1}$$

where τ denotes the full trajectory including the $\langle\texttt{plan}\rangle$ segment and all role outputs, x denotes the input sample, and $\mathcal{O}$ is the set of output tokens to be optimized.

After training, URAFM is able to perform controllable multi-path reasoning over the fixed path set. In the **adaptive planning** mode, given an input $x = (q, i)$, the model first predicts a path

$$\hat{p} \sim \pi_\theta(\cdot \mid x, \mathcal{P}),$$

that is, it first generates the $\langle\texttt{plan}\rangle \cdots \langle\texttt{/plan}\rangle$ segment, and then produces the subsequent role trajectory and final label along the predicted path. Meanwhile, because path prefixes are explicitly modeled during training, URAFM also supports **fixed-path prefix control**: we can directly specify a path prefix and force the model to follow that path for subsequent reasoning. The former is used for sample-level path selection, while the latter provides a unified and controllable interface for path analysis and multi-path aggregation.

3.3 Multi-Path Expert Voting

After URAFM has learned controllable multi-path reasoning, we further exploit the complementarity across different paths by introducing multi-path expert voting at inference time. The core idea is that different paths correspond to different ways of organizing evidence and different reasoning biases, so aggregating multiple paths can mitigate the local bias of any single path.

Specifically, given an input sample $x = (q, i)$, we use fixed-path prefix control to force the model to reason along multiple predefined paths. For each path p_k, the model produces a corresponding prediction

$$\hat{y}_k = f_\theta(x, p_k).$$

When the top N_{expert} paths are selected for aggregation, we treat them as a set of **path experts** and obtain the final label by majority voting:

$$\hat{y} = \arg\max_{c \in \mathcal{C}} \sum_{k=1}^{N_{\text{expert}}} \mathbf{1}(\hat{y}_k = c),$$

where $\mathcal{C}$ denotes the label set.

Unlike conventional repeated-sampling voting, the gain in our method does not primarily come from randomness within the same path, but from the **structural differences and complementarity across different paths**. Therefore, multi-path expert voting can be viewed as a direct way to further exploit URAFM's learned multi-path capability at inference time.

4 Experiments

4.1 Dataset and Evaluation Metrics

Our dataset is an internal multilingual food-delivery relevance dataset constructed from online click logs and rigorous human verification. Each sample consists of a query, an item, and a relevance label. The data covers multiple languages, including Chinese and Arabic, with a 1:1 ratio between Hong Kong (Chinese) and Saudi Arabia (Arabic) region data.

To ensure diverse coverage across query difficulty levels, samples are constructed via **uniform stratified sampling across 9 groups** formed by cross-categorizing query frequency (3 tiers based on online click logs) and query-item BGE-M3 semantic similarity (3 tiers). The label distribution is balanced at 1:1:1 across labels 0, 1, and 2.

We formulate relevance annotation as a three-way classification task, where labels 2, 1, and 0 denote highly relevant, weakly relevant, and irrelevant, respectively. The **training set** contains 150K samples, whose multi-role structured trajectories are synthesized by an external multi-agent teacher (**Gemini 2.5 Pro**). On the training set, the teacher achieves an agreement of $\mathrm{Acc} \approx 0.80$ with human labels, and only trajectories whose final prediction matches the ground-truth label are retained (per-path retention rate $\approx 79\%$).

We use Accuracy and Weighted-F1 as the primary evaluation metrics. We also report per-label accuracy for all three classes to provide a more fine-grained analysis of model behavior across labels.

4.2 Implementation Details

Unless otherwise specified, all experiments are conducted on Qwen3-8B. We fine-tune the base model with LoRA under supervised learning. At inference time, we evaluate both *adaptive planning* and *fixed-path prefix control*. For the latter, we further perform reasoning along multiple predefined paths and apply majority voting for multi-path expert voting. Unless otherwise stated, the paths used for aggregation are ordered by their performance on the training set, from best to worst. More detailed training and inference hyperparameters are provided in the appendix.

4.3 Main Results

Table 1 presents the main results. Among the general baselines, Qwen3-8B achieves 0.7529 Accuracy and 0.7688 Weighted-F1 in the zero-shot setting, which improve to 0.7823 and 0.7853 after label-only supervised fine-tuning (SFT). BGE-M3 and mBERT achieve 0.7792 / 0.7882 and 0.7760 / 0.7759, respectively. Compared with these single-model baselines, URAFM (Adaptive Planning) reaches 0.8170 Accuracy and 0.8183 Weighted-F1, showing that jointly learning path planning, agent execution, and label prediction within a single model substantially improves relevance judgment.

We further compare URAFM with the conventional external multi-LLM multi-agent workflow. The Qwen3-8B + External Multi-LLM Workflow baseline obtains 0.7926 Accuracy and 0.8041 Weighted-F1, whereas URAFM (Adaptive Planning) achieves better overall performance. This suggests that, compared with distributing agent capabilities across an external collaborative pipeline, unifying them within a single model and learning them through explicit path supervision is more effective for the final task.

To disentangle the two confounding variables in this comparison—the unified single-model structure and the stronger teacher supervision quality—we conduct two targeted control experiments. **URAFM (Weak Teacher / Self-Teacher)** replaces the teacher with untuned Qwen3-8B and obtains 0.7912 Accuracy, which already exceeds Qwen3-8B Label-only SFT (0.7823), demonstrating that the unified multi-role training structure provides an independent contribution beyond pure label supervision. **Teacher LLM (Gemini 2.5 Pro) + External Multi-LLM Workflow** deploys the stronger teacher directly on the test set without any domain fine-tuning, and obtains 0.8014 Accuracy—still lower than URAFM (0.8170). This confirms that domain SFT provides an irreplaceable contribution, especially on fine-grained categories such as Label 1 (0.7641 vs. 0.8168).

Multi-path expert voting further unlocks inference-time aggregation gains: $N = 5$ reaches 0.8319 Acc / 0.8334 F1, and $N = 9$ further improves to 0.8399 / 0.8414, confirming that learned multi-path capability translates into more robust predictions through path-level aggregation.

4.4 Ablation and Analysis

4.4.1 Role Ablation Analysis

Table 2 reports the results of role ablation. Overall, the full URAFM (SFT) setting achieves the best performance, showing that structured multi-role

Exp	Acc	F1
URAFM (SFT)	**0.8170**	**0.8183**
w/o clk	0.8006	0.7999
w/o sem	0.7922	0.7912
w/o review	0.8023	0.8018

Table 2: Ablation results for removing different role contents from the training trajectories. "w/o clk", "w/o sem", and "w/o review" denote masking the corresponding role content while keeping the overall path structure unchanged.

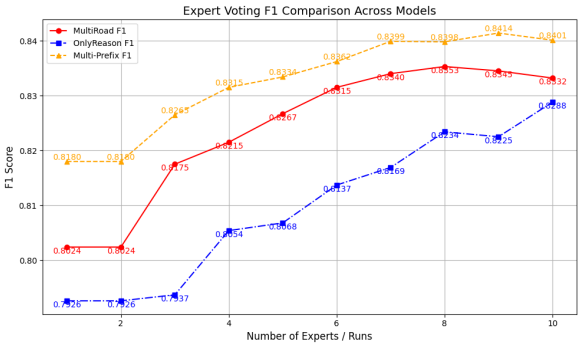

Figure 4: Comparison of different voting strategies under increasing expert/sample budgets ($N = 1$ to 10). **Structured Multi-Path Voting** (ours): aggregates predictions from the top-N fixed-path experts in ranked order. **Adaptive-Path Self-Consistency**: samples N times from URAFM's adaptive planning mode with temperature > 0 and takes majority vote. **Label-Only Self-Consistency**: samples N times from a label-only SFT model and takes majority vote. **OnlyReason**: uses only the `reason` path without any multi-role supervision. **Multi-Prefix (Unbalanced)**: uses fixed-path prefix control but with paths selected without path-balanced training. Structured multi-path voting consistently outperforms all baselines, indicating that the gain mainly comes from aggregating complementary path experts with deterministic structural diversity, rather than merely increasing the number of random samples.

supervision plays an important role in model effectiveness. Among the three ablations, removing `sem` causes the largest performance drop, indicating that the Semantic Matching Agent provides the most critical supervision signal for relevance judgment. Removing `clk` and `review` also degrades performance, but to a smaller extent. Here, `clk` mainly supports click-intent inference, while `review` mainly serves as a mechanism for error correction and consistency enhancement. Overall, the relative importance of the three roles is: `sem` > `clk` > `review`.

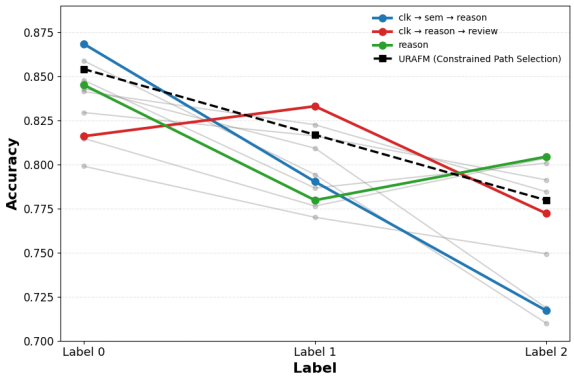

Figure 5: Per-label accuracy of different fixed reasoning paths. Different paths exhibit distinct strengths on Label 0, Label 1, and Label 2, indicating that they are not redundant repetitions of the same reasoning process, but complementary path experts.

4.4.2 Multi-Path Expert Voting Analysis

Figure 4 presents the results of multi-path expert voting under different numbers of experts. Overall, as more paths are included in aggregation, both Accuracy and Weighted-F1 consistently improve, indicating that different paths indeed contain complementary information that can be effectively aggregated.

Specifically, as the number of experts increases from 1 to 9, Accuracy improves from 0.8162 to 0.8399, and Weighted-F1 improves from 0.8180 to 0.8414. This shows that path-level aggregation can effectively improve the stability and accuracy of final predictions. When the number of experts further increases to 10, the overall metrics fluctuate slightly but remain strong, suggesting that while additional paths can provide richer evidence, they may also introduce extra noise. Overall, under the current setting, aggregation with $N = 9$ yields the best performance.

4.4.3 Path Analysis

Figure 5 shows the per-label accuracy of different fixed paths. Overall, different paths exhibit clear differences in both overall performance and class-wise behavior, indicating that path selection itself has a substantial impact on the final prediction. More importantly, the relative ranking of paths differs across the three labels, suggesting that they are not homogeneous variants of one another, but conditional predictors with stable specialization patterns across labels.

More specifically, different paths are advantageous for different labels: `clk` → `sem` →

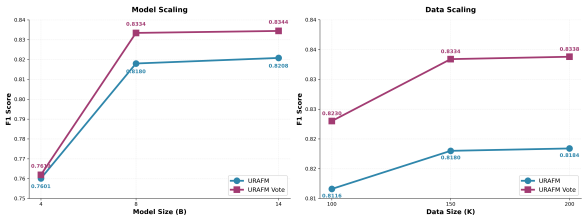

Figure 6: Scaling law comparison for URAFM: F1 scores under different model sizes (4B/8B) and training set sizes (100k/150k/200k), with and without multi-path expert voting. Both larger models and more data consistently improve performance, while multi-path voting provides an additional boost at all scales.

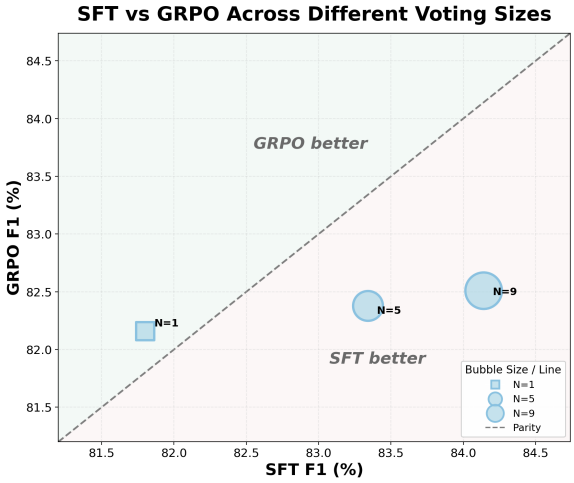

Figure 7: Comparison of SFT and GRPO across different voting sizes N in terms of Weighted-F1. The x-axis denotes SFT performance and the y-axis denotes GRPO performance; larger bubbles indicate larger N. Points above the diagonal indicate that GRPO performs better, while points below the diagonal indicate that SFT performs better. GRPO is slightly better at $N = 1$, but SFT scales more effectively with larger N and achieves the best performance at $N = 9$.

`reason` performs best on Label 0 Accuracy, `clk` $\rightarrow$ `reason` $\rightarrow$ `review` performs best on Label 1 Accuracy, and the `reason` path achieves the highest result on Label 2 Accuracy. This shows that different paths differ substantially in evidence source, step composition, and decision bias, and can therefore be naturally viewed as complementary path experts.

4.4.4 Scaling Law Analysis

As shown in Figure 6, both model scaling and data scaling exhibit clear diminishing returns. For URAFM, the gain from 4B to 8B is +0.0579, and +0.0064 from 100K to 150K but only +0.0004 from 150K to 200K. Meanwhile, multi-path voting consistently provides positive gains across all settings, with improvements of +0.0018/+0.0154 under model scaling and +0.0114/+0.0154/+0.0154 under data scaling. These results indicate a clear diminishing-return trend for both model scaling and data scaling, while multi-path voting consistently yields positive gains across all settings.

4.4.5 Different Voting Gains of SFT and GRPO

Figure 7 compares SFT and GRPO under different voting sizes. At $N = 1$, GRPO slightly outperforms SFT, indicating that reinforcement learning improves single-path decision quality. However, this advantage diminishes when multiple path experts are aggregated. SFT surpasses GRPO at $N = 5$ and achieves the best result at $N = 9$, improving from 0.8180 to 0.8414 in Weighted-F1. In contrast, GRPO only increases from 0.8216 to 0.8251.

This gap suggests that SFT better preserves path heterogeneity under path-balanced supervision, making different paths more complementary for voting. By contrast, GRPO tends to concentrate on a few high-reward reasoning patterns, which benefits single-path inference but limits the gains from multi-path expert voting.

Limitations

URAFM is limited to 10 predefined paths, constraining flexibility for unseen patterns. It also inherits teacher noise, and `clk` is prone to click-log exposure bias. Multi-path voting increases inference cost, and domain generalization remains future work.

5 Conclusion

We proposed **URAFM**, which distills multi-role reasoning from external multi-agent workflows into a single LLM via path-balanced SFT over 10 predefined paths, eliminating communication overhead while preserving role boundaries and path heterogeneity. We further introduced **multi-path expert voting** that exploits deterministic structural diversity across paths for more robust predictions. URAFM achieves 0.8170 Acc / 0.8183 F1 (+3.47/+3.30 over 8B label-only SFT), improving to 0.8399 / 0.8414 with $N = 9$ voting, confirming that unified multi-role learning and structured path diversity are complementary and parameter-efficient sources of gain.

References

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, and 1 others. 2023. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *The Twelfth International Conference on Learning Representations*.

Chenhe Dong, Shaowei Yao, Pengkun Jiao, Jianhui Yang, Yiming Jin, Zerui Huang, Xiaojiang Zhou, Dan Ou, Haihong Tang, and Bo Zheng. 2025. Taosr1: The thinking model for e-commerce relevance search. *arXiv preprint arXiv:2508.12365*.

Bingzheng Gan, Yufan Zhao, Tianyi Zhang, Jing Huang, Li Yusu, Shu Xian Teo, Changwang Zhang, and Wei Shi. 2025a. Master: A multi-agent system with llm specialized mcts. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 9409–9426.

Guobing Gan, Kaiming Gao, Li Wang, Shen Jiang, and Peng Jiang. 2025b. Hcmrm: A high-consistency multimodal relevance model for search ads. In *Companion Proceedings of the ACM on Web Conference 2025*, pages 201–210.

Yulu Gan and Phillip Isola. 2026. Neural thickets: Diverse task experts are dense around pretrained weights. *arXiv preprint arXiv:2603.12228*.

Mengkang Hu, Yuhang Zhou, Wendong Fan, Yuzhou Nie, Bowei Xia, Tao Sun, Ziyu Ye, Zhaoxuan Jin, Yingru Li, Qiguang Chen, and 1 others. 2025. Owl: Optimized workforce learning for general multi-agent assistance in real-world task automation. *arXiv preprint arXiv:2505.23885*.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in neural information processing systems*, 36:51991–52008.

Weizhen Li, Jianbo Lin, Zhuosong Jiang, Jingyi Cao, Xinpeng Liu, Jiayu Zhang, Zhenqiang Huang, Qianben Chen, Weichen Sun, Qiexiang Wang, and 1 others. 2025a. Chain-of-agents: End-to-end agent foundation models via multi-agent distillation and agentic rl. *arXiv preprint arXiv:2508.13167*.

Zichong Li, Xinyu Feng, Yuheng Cai, Zixuan Zhang, Tianyi Liu, Chen Liang, Weizhu Chen, Haoyu Wang, and Tuo Zhao. 2025b. Llms can generate a better answer by aggregating their own responses. *arXiv preprint arXiv:2503.04104*.

Hong Liu, Saisai Gong, Yixin Ji, Kaixin Wu, Jia Xu, and Jinjie Gu. 2024. Boosting llm-based relevance modeling with distribution-aware robust learning. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 4718–4725.

Navid Mehrdad, Hrushikesh Mohapatra, Mossaab Bagdouri, Prijith Chandran, Alessandro Magnani, Xunfan Cai, Ajit Puthenputhussery, Sachin Yadav, Tony Lee, ChengXiang Zhai, and 1 others. 2024. Large language models for relevance judgment in product search. *arXiv preprint arXiv:2406.00247*.

Tian Tang, Zhixing Tian, Zhenyu Zhu, Chenyang Wang, Haiqing Hu, Guoyu Tang, Lin Liu, and Sulong Xu. 2025. Lref: A novel llm-based relevance framework for e-commerce search. In *Companion Proceedings of the ACM on Web Conference 2025*, pages 468–475.

Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D Nguyen. 2025. Multi-agent collaboration mechanisms: A survey of llms. *arXiv preprint arXiv:2501.06322*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, and 1 others. 2024. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First conference on language modeling*.

Dezhi Ye, Jie Liu, Junwei Hu, Jiabin Fan, Bowen Tian, Haijin Liang, and Jin Ma. 2025a. Applying large language model for relevance search in tencent. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 5171–5181.

Rui Ye, Xiangrui Liu, Qimin Wu, Xianghe Pang, Zhenfei Yin, Lei Bai, and Siheng Chen. 2025b. X-mas: Towards building multi-agent systems with heterogeneous llms. *arXiv preprint arXiv:2505.16997*.

Kamer Ali Yuksel, Thiago Castro Ferreira, Mohamed Al-Badrashiny, and Hassan Sawaf. 2025. A multi-ai agent system for autonomous optimization of agentic ai solutions via iterative refinement and llm-driven feedback loops. In *Proceedings of the 1st Workshop for Research on Agent Language Models (REALM 2025)*, pages 52–62.

Gang Zhao, Ximing Zhang, Chenji Lu, Hui Zhao, Tianshu Wu, Pengjie Wang, Jian Xu, and Bo Zheng. 2025. Explainable llm-driven multi-dimensional distillation for e-commerce relevance learning. In *Companion Proceedings of the ACM on Web Conference 2025*, pages 631–640.

He Zhu, Tianrui Qin, King Zhu, Heyuan Huang, Yeyi Guan, Jinxiang Xia, Yi Yao, Hanhao Li, Ningning Wang, Pai Liu, and 1 others. 2025. Oagents: An empirical study of building effective agents. *arXiv preprint arXiv:2506.15741*.

Path	Reasoning Path
p_1	reason → review
p_2	sem → clk → reason
p_3	clk → sem → reason → review
p_4	clk → reason
p_5	reason
p_6	sem → clk → reason → review
p_7	sem → reason → review
p_8	sem → reason
p_9	clk → reason → review
p_{10}	clk → sem → reason

Table 3: The 10 predefined valid reasoning paths used in URAFM. Each path defines a valid execution order of role-specialized reasoning steps. During both training and inference, path planning is restricted to selecting one path from this fixed set.

A Additional Method Details

A.1 Predefined Valid Reasoning Paths

In our framework, path planning is restricted to selecting one path from a fixed set of predefined valid reasoning paths, rather than generating arbitrary agent sequences. Each path specifies a valid execution order of the role-specialized steps. Table 3 lists the 10 predefined reasoning paths used in this work.

A.2 Detailed Agent Definitions

Our multi-agent framework consists of five agents: planning, click-intent inference, semantic matching, relevance annotation, and auditing reflection. Among them, the planning agent does not directly predict the relevance label. Instead, it selects one path from a predefined set of valid reasoning paths and encodes it within the ⟨plan⟩ tag. For example,

$$\texttt{clk} \rightarrow \texttt{sem} \rightarrow \texttt{reason} \rightarrow \texttt{review}$$

indicates that the model sequentially performs click-intent inference, semantic matching, relevance annotation, and auditing reflection.

Click-Intent Agent. This agent retrieves the highest-CTR shop names and SPU names from click logs as query neighbors, and uses these click-based signals to infer the user's latent search intent:

$$\begin{aligned} n(q) &= \mathrm{RetrieveTopCTRNeighbors}(q), \\ o_t^{\text{clk}} &= f_\theta(q, n(q)). \end{aligned}$$

Here, $n(q)$ denotes the click-log-based query neighbors.

Semantic Matching Agent. This agent extracts key attributes from both the query and the item, such as ingredients, preparation style, form, category, flavor, and brand, and analyzes their matching relationship across these dimensions:

$$o_t^{\text{sem}} = f_\theta(\text{query features}, \text{item features}).$$

Annotation Agent. This agent outputs the understanding of both the query and the item, conducts relevance analysis based on the available evidence, identifies the atomic rule used for reasoning, and produces the final relevance judgment:

$$o_t^{\text{reason}} = f_\theta(\text{query}, \text{item}, \text{context}).$$

Auditing Reflection Agent. This agent revisits the previous reasoning process, identifies potential flaws or inconsistencies, and corrects errors based on relevance rules and common error patterns:

$$o_t^{\text{review}} = f_\theta(\text{label}, \text{query}, \text{item}, \text{rationale}).$$

Importantly, these agents are not used as the final deployed external multi-agent system. Instead, they serve as a structured supervision mechanism that provides URAFM with multi-step reasoning trajectories with explicit agent boundaries and step order.

A.3 Structured Trajectory Construction and Path Constraints

Given an input sample $x = (q, i)$, we first select a path p from the predefined valid path set

$$\mathcal{P} = \{p_1, p_2, \ldots, p_{10}\},$$

and then perform multi-step reasoning according to the agent order specified by that path. The resulting multi-agent reasoning process can be represented as a trajectory

$$\tau = \{(S_t, \phi_t, o_t)\}_{t=1}^{T}, \tag{2}$$

where S_t denotes the reasoning state at step t, ϕ_t denotes the agent executed at that step, and o_t denotes the output of that agent. Agent selection is jointly constrained by the path and the current state:

$$\begin{aligned} p &\sim \pi_{\text{plan}}(\cdot \mid x, \mathcal{P}), \\ S_t &= \Gamma(S_{t-1}, o_{t-1}), \\ \phi_t &\sim \pi(\cdot \mid S_t, p, \mathcal{A}(S_t)), \\ o_t &= \Phi(\phi_t), \end{aligned} \tag{3}$$

where $\mathcal{A}(S_t)$ denotes the set of agents that are allowed to be activated under state S_t.

In our setting, agent activation follows explicit ordering constraints. For example, `sem` and `clk` must be executed before `reason`, while `review` must be executed after `reason`. For the `clk` step, the state update depends not only on the original query but also on the query neighbors retrieved from click logs. Its function is therefore essentially click-neighbor-based intent inference rather than generic context supplementation.

This trajectory can be further mapped into a supervised format with explicit path labels:

$$\tau = \langle \texttt{plan} \rangle p \langle \texttt{/plan} \rangle \oplus \langle \texttt{trajectory} \rangle,$$

where $p \in \mathcal{P}$, for example,

$$p = \texttt{sem} \rightarrow \texttt{clk} \rightarrow \texttt{reason} \rightarrow \texttt{review}.$$

Importantly, $\langle \texttt{plan} \rangle$ does not represent open-ended generation of arbitrary paths. Instead, it is a **restricted selection from 10 predefined valid paths**. This design ensures controllability of the reasoning process and avoids the search-space explosion caused by open-ended path generation.

A.4 Path Design Principles

The 10 predefined valid paths are designed according to three guiding principles:

1. **Role ordering constraints.** Certain roles have fixed precedence: `sem` and `clk` must precede `reason`, which in turn must precede `review`. This reflects the natural flow of evidence collection (semantic and click-based features) before judgment and auditing.

2. **Coverage of diverse evidence combinations.** The path set is designed to cover all meaningful subsets of evidence roles—from the minimal single-step path (`reason`) to the maximal four-step paths incorporating both `sem`, `clk`, `reason`, and `review`—so that the model can learn diverse evidence integration strategies.

3. **Path-level diversity for complementary voting.** Paths are chosen to differ in their starting roles and step compositions, enabling distinct decision biases across paths. This diversity is a prerequisite for effective multi-path expert voting: paths that are too similar would provide redundant rather than complementary evidence.

A.5 Path-Balanced Offline Data Synthesis

After defining the agents and path constraints, we use an external multi-agent teacher to synthesize structured supervision data offline. Specifically, we enumerate 10 valid reasoning paths and construct approximately 15K high-quality structured trajectories for each path, resulting in a path-balanced training corpus of about 150K instances in total. The per-path retention rate (i.e., the fraction of trajectories whose final prediction matches the ground-truth label) is approximately 79%, indicating that the Gemini 2.5 Pro teacher achieves reliable annotation quality across all paths.

Each trajectory contains three parts: an explicit $\langle \texttt{plan} \rangle$ path label, the intermediate outputs of the corresponding agents, and the final relevance label. Both training and inference are restricted to this fixed path set, without considering the generation of new paths outside the set. In this way, we explicitly preserve path heterogeneity and agent boundaries at the data level, enabling the model to learn multiple reasoning patterns within a unified parameter space rather than only fitting a single-path decision pattern.

This path-balanced data construction strategy has two direct benefits. First, it prevents training from being overly biased toward a small number of frequent paths, which helps preserve behavioral differences across paths. Second, it provides a necessary prerequisite for multi-path expert voting, since path aggregation at inference time is only meaningful when different paths have all been sufficiently learned during training.

A.6 Unified Multi-Agent Learning Details

Given the multi-agent structured supervision data, URAFM jointly learns all agents with a single model, rather than assigning different agents to separate models as in conventional external multi-LLM multi-agent workflows. Its reasoning process can be expressed as

$$\begin{aligned} p &\sim P(p \mid x, \mathcal{P}), \\ S_t &= f_\theta(S_{t-1}, \phi_{t-1}, o_{t-1}), \\ \phi_t &\sim P(\phi \mid S_t, p). \end{aligned} \tag{4}$$

This unified formulation has two key advantages. On the one hand, reasoning trajectories from different agents are mapped into the same parameter space, making shared knowledge, cross-agent transfer, and joint optimization toward the final label possible. On the other hand, path prefixes and

agent tags preserve step boundaries, preventing unified learning from degenerating into unstructured mixed reasoning and instead yielding a controllable multi-path reasoning paradigm over a fixed path set.

A.7 Training and Inference Details

All models are trained on 8 A100 GPUs and fine-tuned with LoRA under DeepSpeed ZeRO-3. The LoRA rank is set to 8 and the LoRA alpha is set to 32, applied to all linear layers. The learning rate is uniformly set to 1×10^{-5}, training is performed in bfloat16, and the warmup ratio is 5%. The per-device batch size is 4 with 4 gradient accumulation steps, and training is performed for 1 epoch.

At inference time, we evaluate both adaptive planning and fixed-path prefix control. For multi-path expert voting, unless otherwise specified, the paths used for aggregation are ordered by their performance on the training set from highest to lowest.

B Additional Experimental Analysis

B.1 Interpretation of Path Experts

At inference time, given different path prefixes p_k, the same URAFM effectively corresponds to a set of different conditional predictors:

$$\hat{y}_k = f_\theta(x, p_k).$$

Although these predictors share the same model parameters θ, they differ in step composition, evidence sources, and the order in which information enters the decision process. As a result, the conditional prediction behavior induced by different paths is not identical, but instead reflects distinct decision biases and label specializations.

We therefore refer to these conditional predictors induced by different path prefixes as **path experts**. This notion differs from conventional repeated sampling: repeated sampling mainly relies on randomness within the same path, whereas our method relies on structural differences across paths. Precisely because these path experts are complementary across different labels and sample types, majority voting over them can effectively reduce the local bias of any single path and improve the stability of the final prediction.

B.2 Path Selection Distribution

Figure 8 shows the distribution of path selection on the test set. Under path-balanced supervision,

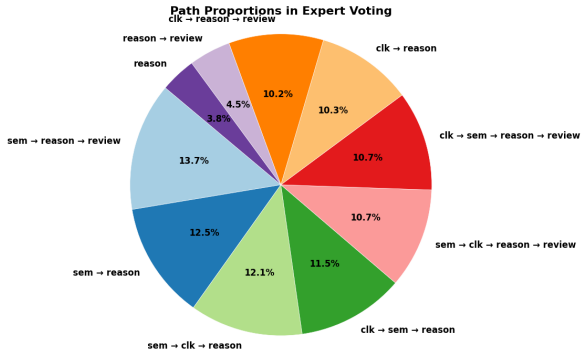

Figure 8: Path selection distribution on the test set after training with path-balanced supervision. The model exhibits relatively balanced path usage, indicating strong path selection and path-following ability over the fixed path set.

URAFM does not collapse to only a few fixed paths, but instead maintains a relatively dispersed path usage pattern. This indicates that the model has learned stable path selection behavior over the predefined valid path set and has strong path-following ability. This phenomenon also provides a necessary condition for subsequent multi-path expert voting: different paths must be sufficiently learned and actually activated before their complementarity can be exploited at inference time.

B.3 Token Consumption Analysis

Beyond performance gains, we further analyze token consumption under different reasoning paradigms, as shown in Table 4. The conventional external multi-LLM multi-agent workflow incurs the highest total token cost, mainly because multiple external agents repeatedly exchange system prompts, user prompts, and intermediate results, resulting in substantial context redundancy and interaction overhead. In contrast, URAFM consolidates the multi-agent capabilities originally distributed across external workflows into a single model, significantly reducing cross-agent communication and repeated prompting overhead.

Among the two URAFM inference modes, adaptive planning has the lowest total token consumption, while fixed-path prefix control incurs slightly higher cost. This indicates that restricted path selection over a fixed set of paths can avoid unnecessary steps while maintaining reasoning quality, thereby reducing overall inference cost. Although fixed-path prefix control remains substantially more efficient than the conventional external multi-LLM multi-agent workflow, its assistant-output cost is

somewhat higher due to longer generated trajectories. From an efficiency perspective, these results further support our core claim: consolidating multi-agent capabilities within a single model not only improves performance but also significantly enhances inference efficiency.

Furthermore, these results suggest that URAFM's efficiency gains do not rely on open-ended path planning, but primarily stem from restricted path selection and unified execution over the predefined valid path set. In other words, even when the path space is explicitly limited to 10 fixed valid paths, the single-model paradigm can still achieve a better trade-off between performance and cost.

B.4 Detailed Results of Multi-Path Expert Voting

Beyond performance gains, we further analyze token consumption under different reasoning paradigms, as shown in Table 4. The conventional external multi-LLM multi-agent workflow incurs the highest total token cost, mainly because multiple external agents repeatedly exchange system prompts, user prompts, and intermediate results, resulting in substantial context redundancy and interaction overhead. In contrast, URAFM consolidates the multi-agent capabilities originally distributed across external workflows into a single model, significantly reducing cross-agent communication and repeated prompting overhead.

Among the two URAFM inference modes, adaptive planning has the lowest total token consumption, while fixed-path prefix control incurs slightly higher cost. This indicates that restricted path selection over a fixed set of paths can avoid unnecessary steps while maintaining reasoning quality, thereby reducing overall inference cost. Although fixed-path prefix control remains substantially more efficient than the conventional external multi-LLM multi-agent workflow, its assistant-output cost is somewhat higher due to longer generated trajectories. From an efficiency perspective, these results further support our core claim: consolidating multi-agent capabilities within a single model not only improves performance but also significantly enhances inference efficiency.

Furthermore, these results suggest that URAFM's efficiency gains do not rely on open-ended path planning, but primarily stem from restricted path selection and unified execution over the predefined valid path set. In other words, even when the path space is explicitly limited to 10 fixed valid paths, the single-model paradigm can still achieve a better trade-off between performance and cost.

B.5 GRPO Implementation Details and Ablation

This section provides complete implementation details for the GRPO experiments and the full ablation results reported in Section B.5.

Base Checkpoint. GRPO training is initialized from the **final checkpoint of path-balanced SFT** rather than directly from the Qwen3-8B base model. This choice is motivated by the fact that the SFT stage has already encoded multi-role reasoning structure and path format into the model parameters. GRPO then performs reward-based optimization on top of this initialization, avoiding the additional cost of RL exploration from scratch while maintaining direct comparability with the SFT baseline.

Path Sampling. GRPO uses **dynamic path sampling**: at each rollout step, the model autonomously generates the `<plan>` segment to select a reasoning path and then completes the subsequent role execution along that path, without any external path prefix constraint. Each prompt generates `num_generations`$= 4$ independent rollouts with temperature $= 1.7$ to encourage path exploration diversity. Training data uses the same path-balanced dataset as SFT, but the path distribution is not enforced during GRPO rollout; paths are determined by the model autonomously.

Reward Function. The GRPO reward function consists of two main components (implemented as an external plugin `reward_plugin_diversity_v3.py` injected into the MS-Swift RLHF training framework):

(1) Correctness Reward:

$$r_{\text{correct}} = \begin{cases} 1.0 & \text{if } \hat{y} = y \\ 0.0 & \text{otherwise} \end{cases} \tag{5}$$

The predicted label within the `<label>` tag is parsed from the rollout output and compared against the ground-truth label.

Setting	System Prompt	User Prompt	Assistant Output	Total
Standalone Agent-Specific Models				
Click-Intent Inference	11,933,522	4,376,092	421,270	16,730,884
Semantic Matcher	13,038,696	4,737,366	1,348,066	18,762,854
Annotation Agent	27,627,237	6,025,402	1,519,074	35,171,713
Auditing Reflection Agent	61,003,899	7,483,476	1,462,476	69,949,851
Workflow-Level Systems				
Multi-LLM Workflow	113,603,354	22,622,336	4,750,886	140,976,576
URAFM (Adaptive Planning)	26,727,327	2,539,561	4,345,553	**33,612,441**
URAFM (Fixed-Path Prefix Control)	26,727,327	2,539,561	5,519,150	34,786,038
URAFM + Multi-Path Voting ($N = 9$)	26,727,327	2,539,561	49,672,350	78,939,238

Table 4: Token consumption comparison across standalone agent-specific models, the conventional external multi-LLM multi-agent workflow, and the unified URAFM inference paradigm. URAFM + Multi-Path Voting ($N = 9$) runs Fixed-Path Prefix Control for each of the 9 paths per sample; its total cost is still substantially lower than the Multi-LLM Workflow per-path overhead.

Method	Init	Path Sampling	Reward	Acc	F1	Acc-0	Acc-1	Acc-2
URAFM-SFT	SFT	path-balanced	CE	0.8170	0.8183	—	—	—
SFT top-1 path	SFT	fixed prefix	CE	0.8175	0.8192	—	—	—
GRPO	SFT init	dynamic	correct. + format	**0.8215**	**0.8216**	—	—	—
GRPO-fixed-prefix-top1	SFT init	fixed prefix	correct. + format	0.8199	0.8215	0.8556	0.8435	0.7606
GRPO + path diversity reward	SFT init	dynamic	correct. + format + diversity	0.8208	0.8217	0.8700	0.8003	0.7921
GRPO w/o format reward	SFT init	dynamic	correctness only	0.8186	0.8184	0.8443	0.8236	0.7879

Table 5: Complete GRPO ablation results. All GRPO variants are initialized from the path-balanced SFT checkpoint. "GRPO-fixed-prefix-top1" enforces the best training-set path prefix during rollout. "GRPO + path diversity reward" adds an explicit path-frequency penalty to mitigate collapse. "GRPO w/o format reward" uses correctness-only reward. GRPO achieves the best single-path accuracy but exhibits path collapse, limiting its multi-path voting gains compared with SFT.

Expert Count	Acc	F1	Acc-0	Acc-1	Acc-2
1	0.8162	0.8180	0.8413	0.8225	0.7845
2	0.8162	0.8180	0.8413	0.8225	0.7845
3	0.8245	0.8265	0.8368	0.8421	0.7945
4	0.8298	0.8315	0.8481	0.8409	0.8002
5	0.8319	0.8334	0.8508	0.8376	0.8071
6	0.8347	0.8362	0.8541	0.8415	0.8083
7	0.8383	0.8399	0.8541	0.8529	0.8077
8	0.8383	0.8398	0.8556	0.8508	0.8083
9	**0.8399**	**0.8414**	0.8565	**0.8590**	0.8041
10	0.8387	0.8401	**0.8612**	0.8568	**0.8083**

Table 6: Detailed results of multi-path expert voting under different numbers of path experts. Best values are shown in bold.

(2) Format Reward:

$$r_{\text{format}} = \begin{cases} 1.0 & \text{if output contains valid <plan>} \\ & \quad \text{+ trajectory structure} \\ 0.0 & \text{otherwise} \end{cases} \tag{6}$$

This checks whether the output contains a valid path declaration, whether role tags appear in the declared order, and whether the output ends with a valid <label> tag.

(3) Path Diversity Reward (used only in the ablation variant "GRPO + path diversity reward"):

$$r_{\text{diversity}}(p) = \frac{1}{|\mathcal{P}|} \cdot \frac{1}{\hat{\pi}(p) + \epsilon} \tag{7}$$

where $\hat{\pi}(p)$ is the empirical frequency of path p in the current rollout batch and ϵ is a smoothing term. Low-frequency paths receive higher rewards

to mitigate path collapse.

The combined reward in the **main GRPO experiment** is:

$$r = r_{\text{correct}} + r_{\text{format}} \tag{8}$$

All other training hyperparameters are consistent with SFT (LoRA rank $= 8$, alpha $= 32$, learning rate $= 1 \times 10^{-5}$, bfloat16, warmup ratio $= 5\%$, 1 epoch) to ensure comparability.

Complete GRPO Ablation Results. Table 5 reports the full ablation results for all GRPO variants. At $N = 1$ (single-path inference), GRPO (dynamic path, correctness + format) achieves the best single-path accuracy of 0.8215. However, this advantage diminishes under multi-path expert voting: the path collapse tendency of GRPO limits its scaling behavior with N, whereas SFT, trained with path-balanced supervision, better preserves path heterogeneity and scales more effectively. The GRPO-fixed-prefix-top1 variant, which enforces the best-performing path prefix during rollout, does not consistently outperform standard GRPO even on single-path metrics, indicating that the performance gain is not solely attributable to path selection. The GRPO + path diversity reward variant partially alleviates path collapse but introduces a tension with correctness reward optimization, resulting in only marginal improvements.

To provide a more fine-grained analysis of the gains from multi-path expert voting, we report the full results under different numbers of experts in Table 6. Overall, as more path experts are included in aggregation, the model performance shows a steady upward trend, improving from 0.8162 Accuracy / 0.8180 Weighted-F1 with 1 expert to 0.8399 / 0.8414 with 9 experts. This further confirms that different paths within the fixed path set indeed contain complementary information that can be effectively exploited, and that multi-path aggregation can mitigate the local bias of any single path.

From the class-wise results, the optimal number of experts is not identical across labels. With 9 experts, the model achieves the best overall Accuracy, Weighted-F1, and Label 1 Accuracy, while with 10 experts, it achieves the best Label 0 Accuracy and Label 2 Accuracy. This suggests that, although incorporating more paths provides richer evidence, additional paths may also introduce noise. Therefore, more experts do not necessarily lead to better overall performance. Overall, under the current setting, 9 path experts provide the best overall trade-off.

Region	Acc	F1	Acc-0	Acc-1	Acc-2
Chinese (HK)	0.8467	0.8475	0.9045	0.8656	0.7695
Arabic (SA)	0.7871	0.7884	0.8032	0.7647	0.7933
Overall	0.8170	0.8183	0.8541	0.8150	0.7814

Table 7: Per-language performance of URAFM (Adaptive Planning) on the test set. HK denotes the Hong Kong Chinese subset and SA denotes the Saudi Arabia Arabic subset. Overall results are computed across both subsets.

C Per-Language Performance Analysis

C.1 Per-Language Results

Table 7 reports the per-language performance of URAFM (Adaptive Planning) on the test set. The dataset covers two regions: Hong Kong (Chinese, HK) and Saudi Arabia (Arabic, SA), with a 1:1 regional split.

The Chinese (HK) subset achieves 0.8467 Accuracy and 0.8475 Weighted-F1, while the Arabic (SA) subset achieves 0.7871 and 0.7884, respectively. The performance gap between the two languages (HK $>$ SA by approximately 6 points) is consistent with the pretraining data scale differences between Chinese and Arabic in the base model. Notably, both languages exhibit stable per-class accuracy distributions across all three labels, confirming that URAFM generalizes effectively in a cross-lingual setting rather than overfitting to a single language.

C.2 Multilingual Query–Item Examples

We will add anonymized Arabic and Chinese query–item examples illustrating how URAFM selects different reasoning paths for samples from each language. These examples are intended to demonstrate that the path planning mechanism adapts to linguistic and cultural differences in food-delivery search behavior (e.g., ingredient naming conventions, dish category taxonomies), and that multi-role structured reasoning remains interpretable across languages.

D Prompt Templates

RAgents Click-based Intent Inference Prompt

You are an expert in click-based intent inference. Your task is to infer the user's main search intent based on the user query and the click-based neighbors retrieved from click logs, and provide auxiliary evidence for subsequent relevance judgment.

Input:

- Query: user search query
- Shop Neighbors: high-CTR historical shop neighbors
- Item Neighbors: high-CTR historical item neighbors

Requirements:

- Identify what the user is mainly searching for.
- Briefly summarize the core intent and possible auxiliary constraints.
- Summarize the intent signals provided by the click-based neighbors.
- The output must be a single valid JSON object.

Output Format (JSON):

```
{
  "core_intent": "...",
 "auxiliary_intents": ["...", "..."],
 "target_type": "item|shop|brand|...",
  "click_signal_summary": "...",
  "intent_summary": "..."
}
```

RAgents Semantic Matching Agent Prompt

You are an expert in keyword extraction and matching. Your task is to extract key features from a user query (Query) and an item (Item), and analyze their match across several dimensions.

Input:

- Query: user's search query
- Item: item information, e.g., SPU fields; can be a dish or shop

Output:

- Query and Item keywords
- Matching analysis across key dimensions: ingredient, cooking method, form, category, taste, special requirement, regional feature, brand

Example: {examples}
Output Format (JSON):

```
{
  "query_core_word": "...",
  "item_core_word": "...",
 "core_word_match": "yes|no|partial",
  "query_keywords": ["...", "..."],
  "item_keywords": ["...", "..."],
  "matching_analysis": {
    "ingredient": "...",
    "cooking_method": "...",
    "shape": "...",
    "category": "...",
    "taste": "...",
    "special_requirement": "...",
    "regional_feature": "...",
    "brand": "..."
  },
  "overall_matching": "...",
  "reasoning": "..."
}
```

RAgents Relevance Annotation Prompt

You are an expert in query-item relevance annotation. Your task is to analyze a user query (Query) and an item (Item) and provide a professional relevance judgment based on annotation rules.

Core Tasks:

- Understand the core intent of the Query.
- Analyze key features and components of the Item.
- Compare the Query and Item to determine relevance.
- Reference the applicable atomic scoring rule.
- Assign a final relevance label (0 = irrelevant, 1 = partial match, 2 = exact match).

Examples: {examples}
Output Requirements (JSON):

```
{
  "reasoning": {
    "query_understanding": "...",
    "item_understanding": "...",
    "detailed_analysis": "...",
    "atomic_rule": "...",
    "final_label": "..."
  },
  "label": 0|1|2,
  "confidence": 0.0-1.0
}
```

Important:

- Return a single JSON object; no extra text.
- $label$ must be an integer: 0, 1, or 2.
- $confidence$ must be a float between 0 and 1.

- $atomic_rule$ must be one of the seven rules: Exact Match, Core Satisfied, Category Included, Reasonable Substitution, Combo/Set, Functional Similarity, Completely Different.
- $reasoning$ must include five subfields: $query_understanding$, $item_understanding$, $detailed_analysis$, $atomic_rule$, $final_label$.
- Ensure valid JSON format, directly parsable.

RAgents Audit & Reflection Agent Prompt

You are a "Search Relevance Audit Agent". Review annotated data to ensure compliance with relevance rules. Only base judgments on input fields; do not use external knowledge. Lines will automatically wrap if they exceed the page or column width.

Input: Each record must contain at least $search_word$, spu_local, spu_lang, $shop_name_multilang$, $label$; optional: spu_local_sw, $shop_name_multilang_sw$ (only for intent inference).

Task: Reconstruct the correct label (0/1/2), compare with existing $label$, and output is_pass and $fail_reasons$.

Relevance Labels:

- 2 = Exact match
- 1 = Partial match
- 0 = Irrelevant

Common Error Categories (Simplified):

- $E01$ – Target object confusion (SPU vs POI)
- $E02$ – Incorrect scoring (using CTR or auxiliary fields)
- $E03$ – Intent misinterpretation
- ... (other errors omitted for brevity)

Output Format (JSON):

```
{
  "is_pass": true|false,
  "fail": ["E01", "E02", ...],
  "current_label": 0|1|2,
  "reason_analysis": "..."
}
```

URAFM Multi-Agent Reasoning Chain Prompt

This prompt guides the multi-agent reasoning process for relevance annotation. Use only the 5 functions: $plan$, clk, sem, $reason$, $review$.

Label Rules:

- 2 = Exact match: SPU fully or almost fully satisfies the Query
- 1 = Partial match: SPU partially satisfies or is a reasonable substitute
- 0 = Irrelevant: SPU does not satisfy the Query

Function Descriptions:

- **plan**: Suggest a reasoning path for the sample, e.g., <plan>clk, sem, reason, review</plan>
- **clk**: Retrieve historically relevant shop/SPU names to assist intent understanding
- **search**: Obtain external query-related information
- **sem**: Extract key features from Query and Item and perform multi-dimensional matching analysis (ingredient, method, form, category, taste, special requirement, regional, brand)
- **reason**: Generate reasoning explaining the relationship between Query and Item
- **review**: Audit annotated labels for compliance and flag failures with standardized codes
- **label**: Output the final relevance label $(0/1/2)$

Common Error Categories (Simplified):

- $E01$ – Target object confusion (SPU vs POI)
- $E02$ – Incorrect scoring (using CTR or auxiliary fields)
- $E03$ – Intent misinterpretation
- ... (other errors omitted)

Output Formats:

<sem> output example:

```
<sem> {
  "query_core_word": "...",
  "item_core_word": "...",
  "core_word_match": "yes|no|partial",
  "core_word_match_reason": "...",
  "reasoning": "..."
} </sem>
```

<reason> output example:

```
<reason> {
  "reasoning": {
    "query_understanding": "...",
    "item_understanding": "...",
    "detailed_analysis": "...",
    "atomic_rule": "...",
    "final_label": "..."
  },
  "label": 0|1|2,
  "confidence": 0.0-1.0
} </reason>
```

<review> output example:

```
<review> {
  "is_pass": "true|false",
  "current_label": 0|1|2,
  "reason_analysis": "..."
} </review>
```

<label> output:

```
<label>0|1|2</label>
```

E Case Study: Full Multi-Agent Reasoning Chain Output Example

The following example illustrates a complete URAFM multi-agent reasoning chain output under the path `clk` → `sem` → `reason` → `review`. The input sample is a food-delivery relevance annotation instance where the query is *"Tomato and Egg Stir-fry"* and the candidate item is *"C15 Shrimp and Egg Stir-fry"* from a Chinese restaurant. The expected output label is 1 (partial match / reasonable substitute).

Input Fields.

Query: Tomato and Egg Stir-fry ()

Item: C15 Shrimp and Egg Stir-fry (C15)

Shop Name: Wu Jiang Liu Fu (Chinese Traditional Beijing-Sichuan-Shanghai Cuisine)

Shop Category: Full-service Restaurant / Cuisine > Chinese > Sichuan

Reasoning Chain Output.

Path Planning. <plan> clk → sem → reason → review </plan>

Click-Intent Agent (clk). <clk> {
"core_intent": "Looking for a classic home-style dish made with tomato and egg stir-fried together",
"auxiliary_intents": ["egg stir-fry dishes"],
"target_type": "item",
"click_signal_summary": "SPU neighbors under the same query include 'Tomato and Egg Stir-fry' (variant spelling); POI neighbor is 'Superman Roast Goose'. Click neighbors concentrate on egg stir-fry dishes, indicating a clear and focused user intent.",
"intent_summary": "The user is explicitly searching for the classic home-style tomato-and-egg stir-fry dish, with core intent centered on the tomato + egg combination."
} </clk>

Semantic Matching Agent (sem). <sem> {
"query_core_word": "Tomato and Egg Stir-fry",
"item_core_word": "Shrimp and Egg Stir-fry",
"core_word_match": "partial",
"core_word_match_reason": "Both share 'egg stir-fry'; cooking method (stir-fry) and dish form are identical, but the main ingredient differs: tomato (Query) vs. shrimp (Item).",
"matching_analysis": {
"ingredient": "Partial - both contain egg, but main ingredient differs (tomato vs. shrimp)",
"cooking_method": "Full match - both use stir-fry",
"shape": "Match - both belong to the egg stir-fry dish form",
"category": "Match - both are Chinese cuisine",
"taste": "Partial - Query is sour-sweet-savory; Item is savory-umami",
"special_requirement": "None",
"regional_feature": "No significant distinction",
"brand": "None"
},
"overall_matching": "partial",
"reasoning": "Query and Item are highly consistent in cooking method, dish form, and category. There is a moderate taste difference and partial ingredient overlap. Overall semantic matching is medium."
} </sem>

Annotation Agent (reason). <reason> {
"reasoning": {
"query_understanding": "The user intends to

find the classic home-style dish stir-fried with tomato and egg as the two main ingredients.",
    "item_understanding": "'C15 Shrimp and Egg Stir-fry' uses shrimp and egg as main ingredients; it belongs to the egg stir-fry series but contains no tomato. The shop is categorized under Sichuan cuisine.",
    "detailed_analysis": "Both share the 'egg stir-fry' cooking form, but the core ingredients differ significantly: tomato (Query) vs. shrimp (Item). The Item can be regarded as a reasonable substitute within the egg stir-fry category and partially satisfies the user's need, but cannot fully meet the explicit tomato + egg requirement.",
    "atomic_rule": "Reasonable Substitution",
    "final_label": "1"
  },
  "label": 1,
  "confidence": 0.85
} </reason>

Auditing Reflection Agent (review).
<review> {
  "is_pass": "true",
  "current_label": 1,
  "reason_analysis": "Query 'Tomato and Egg Stir-fry' and SPU 'Shrimp and Egg Stir-fry' both belong to the egg stir-fry category, but their main ingredients differ (tomato vs. shrimp). This constitutes a reasonable substitution within the same dish family and partially satisfies the search need. Label = 1 is consistent with the relevance annotation rules; no correction required."
} </review>

Final Label. <label>1 </label>