AgentRouter: A Knowledge-Graph-Guided LLM Router for Collaborative Multi-Agent Question Answering

Zheyuan Zhang[1*], Kaiwen Shi[1*], Zhengqing Yuan[1], Zehong Wang[1], Tianyi Ma[1], Keerthiram Murugesan [3], Vincent Galassi[1], Chuxu Zhang[2], Yanfang Ye[1†]
[1]University of Notre Dame, [2]University of Connecticut, [3]IBM Research,
[*]Equal Contribution [†]Corresponding Author
{zzhang42, yye7}@nd.edu,

Abstract

Large language models (LLMs) and agent-based frameworks have advanced rapidly, enabling diverse applications. Yet, with the proliferation of models and agentic strategies, practitioners face substantial uncertainty in selecting the best configuration for a downstream task. Prior studies show that different agents and backbones exhibit complementary strengths, and that larger models are not always superior, underscoring the need for adaptive routing mechanisms. Existing approaches to agent routing, however, often emphasize cost efficiency while overlooking the fine-grained contextual and relational structure inherent in QA tasks. In this paper, we propose **AgentRouter**, a framework that formulates multi-agent QA as a knowledge-graph–guided routing problem supervised by empirical performance signals. Specifically, we convert QA instance into a knowledge graph that jointly encodes queries, contextual entities, and agents, and then train a heterogeneous graph neural network (GNN) to propagate information across node types and produce task-aware routing distributions over agents. By leveraging soft supervision and weighted aggregation of agent outputs, **AgentRouter** learns principled collaboration schemes that capture the complementary strengths of diverse agents. Extensive experiments demonstrate that our framework consistently outperforms single-agent and ensemble baselines, while generalizing across benchmarks and LLM backbones. These results highlight the effectiveness and robustness of graph-supervised multi-agent routing for question answering. Our code repo is available here.

1 Introduction

Large language models (LLMs) (Ye et al., 2025) have quickly established themselves as versatile learners, demonstrating strong performance in reasoning, comprehension, and natural language generation. Their rapid advancement has significantly influenced both academic research and real-world applications, spanning areas from scientific exploration to software engineering (Kojima et al., 2022; Ouyang et al., 2022). Building on this progress, LLM-driven agents have gained prominence for their ability to plan, interact, and solve complex tasks with limited human oversight. These agents extend LLMs into practical pipelines, enabling applications such as program synthesis and repair, retrieval-augmented generation, data analysis, and interactive decision-making (Jimenez et al., 2024; Singh et al., 2025; Guo et al., 2024a; Li et al., 2024a; Ma et al., 2025c; Zhang et al., 2026).

Despite these advances, the rapidly expanding ecosystem of LLMs and agentic strategies introduces a pressing challenge: given a fixed downstream task, practitioners face substantial uncertainty about which model or agent configuration is most effective. Prior studies consistently show that different agents and LLM backbones exhibit task-dependent strengths, and that larger models do not always outperform smaller ones in every scenario (Chen et al., 2024d,a). This heterogeneity underscores that a one-size-fits-all solution is inherently suboptimal, motivating the development of adaptive routing mechanisms.

Major Research Gaps. To address this challenge, recent research has explored LLM selection and routing (Shi et al., 2026; Jiang et al., 2023a; Ong et al., 2025b; Chen et al., 2024b). These studies highlight the potential of adaptive selection, yet important limitations remain. **First,** most prior approaches neglect the rich structural context underlying QA tasks. Many ignore semantic information in the input context altogether, and even works that attempt to incorporate supervised signals (e.g., (Feng et al., 2025)) still fall short in modeling the fine-grained contextual structures that drive effective reasoning. **Second,** much of the literature emphasizes cost efficiency or adaptability across heterogeneous tasks. While valuable, this perspective

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 788–809
July 2-7, 2026

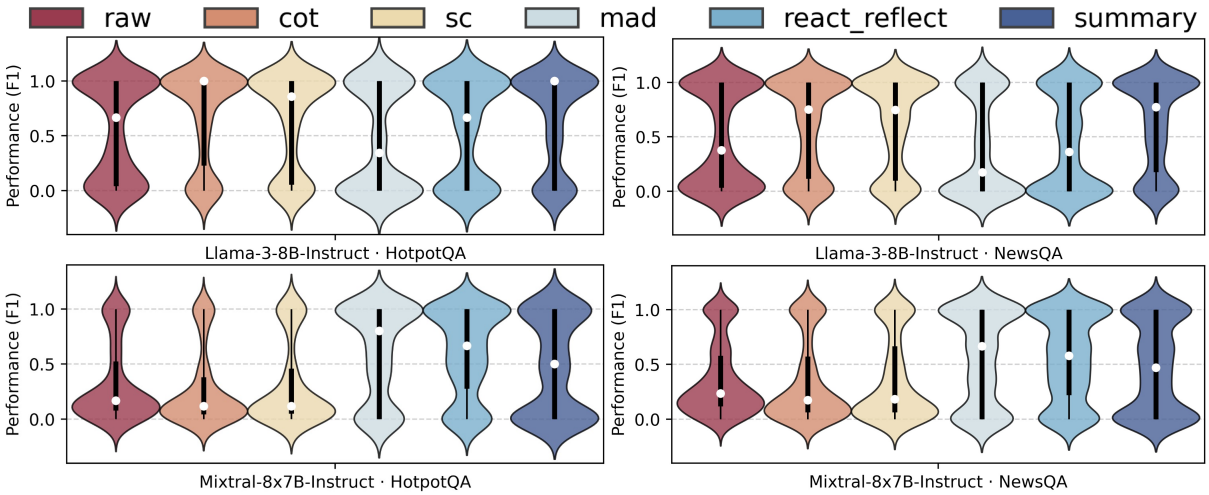

Figure 1: **Our Motivation.** Performance variance across six classical agent designs (raw, CoT, self-consistency, MAD, ReAct-Reflect, and summary), each applied with *the same* prompt template. Results are shown for two LLM backbones (Llama-3-8B-Instruct, Mixtral-8×7B-Instruct) on two benchmarks (HotpotQA, NewsQA). Each violin plot depicts the distribution of F1 scores across test instances, with the white dot indicating the median and the black bar the interquartile range. **The plots highlight agents with the same backbone yield wide and non-overlapping distributions, and the relative ranking of agents varies substantially across tasks and backbones.**

overlooks another prevalent scenario: the downstream task is fixed, but new data continuously arrives, requiring the optimal collaboration scheme beyond the single best agent that maximizes future performance.

To mitigate the aforementioned gaps, we propose **AgentRouter**, a framework that formulates multi-agent Question Answering (QA) as a knowledge-graph based routing problem guided by supervised signals. Our approach proceeds in two stages. First, we construct a knowledge graph in which queries, contextual entities, and agents are jointly represented, with edges encoding lexical, semantic, and relational signals. Prior research has shown that knowledge graphs are particularly beneficial for tasks involving multi-hop reasoning or relational context (Jin et al., 2024; Jiang et al., 2024; Peng et al., 2024), making them a natural fit for routing in complex QA. Second, we propose a heterogeneous graph neural network (GNN) over this knowledge graph setting to propagate information across node types and to produce task-aware routing distributions over agents. The router leverages soft supervision derived from empirical agent performance, learning distributions rather than hard assignments, and generates final predictions through weighted vote aggregation of agent outputs.

Why it matters? By embedding queries, entities, and agents into a unified graph, AGENTROUTER grounds agent selection in the same semantic structures that govern reasoning for QA. By learning collaboration schemes from supervised graph signals, rather than relying on heuristic voting or LLM-based judges, our method adapts to new inputs while effectively capturing the complementary strengths of diverse agents. Extensive experiments confirm that our approach consistently outperforms single-agent and ensemble baselines, while also generalizing across benchmarks and backbones. These results underscore the effectiveness and robustness of graph-supervised multi-agent routing. Our contributions can be summarized as follows:

- **KG-based Formulation of Multi-Agent QA.** We present the first framework that converts multi-agent question answering into knowledge graphs, where nodes represent not only queries and agent prompts but also fine-grained entities and contextual interactions, thereby preserving and exploiting latent semantic dependencies critical for complex QA.

- **Graph-Supervised Collaboration Learning.** We propose a framework that learns an adaptive collaboration scheme across diverse agent designs and LLM backbones, which leverages supervised graph signals instead of LLM-based judges or heuristic ensembling.

- **Robust Empirical Validation.** Extensive experiments show that our collaboration scheme consistently outperforms the best single agent and state-of-the-arts baselines.

2 Problem Formulation

Overarching Goal. We study the problem of designing an optimized LLM-based agent router for a fixed downstream task. Let $\mathcal{A} = \{a_1, \ldots, a_n\}$ denote the pool of available agents (each agent defined by a backbone LLM and an interaction strategy/prompting style), $\mathcal{X}$ the input/query space, and $\mathcal{Y}$ the output space. For $x \in \mathcal{X}$, each agent $a \in \mathcal{A}$ produces a candidate $y_a(x) \in \mathcal{Y}$. Our goal is to learn an optimal weighted combination of agents in $\mathcal{A}$ that maximizes task-level performance in the fixed downstream setting.

Assumption and Empirical Validation. Our formulation builds on the assumption that no single agent or backbone uniformly dominates; rather, their strengths and weaknesses are task- and backbone-dependent. This premise is supported by extensive prior work, which consistently shows that different LLMs or prompting strategies excel in different scenarios. Such heterogeneity has motivated collaborative frameworks, where combining diverse models yields higher overall performance than any single constituent (Chen et al., 2024d,a). We further validate this assumption through our experiments (Figure 1). Specifically, we observe that for a fixed backbone and dataset, agent F1 distributions are wide with non-overlapping confidence intervals, and the agent that performs best on one benchmark is often suboptimal on another. Even under identical prompt templates, performance rankings vary substantially across benchmarks and backbones. These findings substantiate the need for a principled, task-aware routing mechanism rather than a one-size-fits-all collaboration scheme.

Contextual Modeling via Knowledge Graphs. To capture the contextual information required for routing, we represent the interaction space as a knowledge Graph (KG). Formally,

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}), \quad \mathcal{V} = \mathcal{V}_Q \cup \mathcal{V}_A \cup \mathcal{V}_E,$$

where $\mathcal{V}_Q$ denotes query nodes, $\mathcal{V}_A$ agent nodes, and $\mathcal{V}_E$ entity nodes capturing contextual information. Edges encode different types of relationships: query–entity and entity–entity edges capture semantic or entity-level relations derived from the input, while query–agent edges capture agent responses or performance signals. The statistics of the KGs across benchmarks can be seen in Table 1. Within this KG, the routing problem reduces to learning a function f_θ that scores query–agent pairs by propagating signals along graph edges:

$$s(q, a) = f_\theta(q, a; \mathcal{G}),$$

where $s(q, a)$ estimates the utility of including agent a when solving query q for the given task. The router then computes a weighted combination of agents:

$$\hat{y}(q) = \phi(\{\, y_a(q),\, w_a(q) \,:\, a \in \mathcal{A}\}),$$

with weights $w_a(q) \propto \exp(s(q, a))$, and ϕ denoting an aggregation rule such as voting, reranking, or learned fusion. Framing the problem in this way refines the high-level goal of "finding the best collaboration scheme" into the concrete task of learning graph-supervised scores for query–agent pairs, which in turn yield optimized weightings of agents for a fixed downstream task.

Statistics	HotpotQA	NewsQA
Avg. # of Nodes:		
Query	1	1
Agent	24	24
Entity	111.6	46.0
Avg. # of Edges:		
Entity-Entity	331.6	110.34
Agent-Entity	74.64	89.18
Query-Entity	22.96	3.80

Table 1: Comparison of knowledge graph statistics between multi-hop and direct QA benchmarks. Agent and query node number are fixed across benchmarks.

3 Methodology

Now that we have properly formalized the problem, we proceed to detail our proposed framework. As illustrated in Figure 2, we first transform original query–context pairs into knowledge graphs. These graphs serve as a semantically grounded substrate that captures both contextual evidence and agent-specific perspectives. On top of this representation, we introduce RouterGNN, a type-aware heterogeneous GNN trained to infer query–agent compatibility distributions from empirical performance signals. This design enables the router to move beyond static ensembling, learning adaptive collaboration strategies that exploit complementary agent strengths.

3.1 Knowledge Graph Construction

To enable context-aware routing, we convert each QA instance (q, C), with query q and context C,

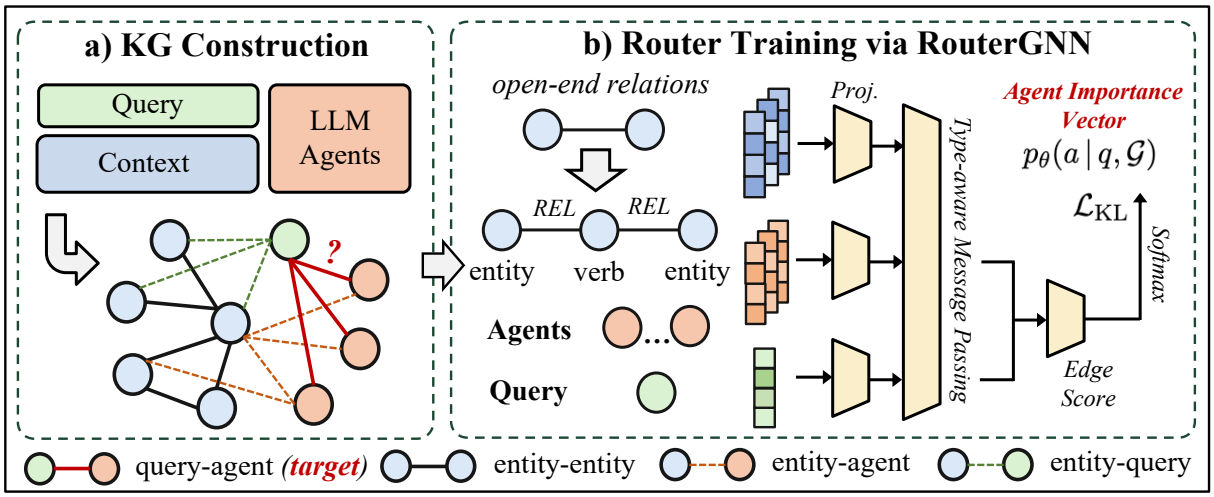

Figure 2: Overview of our proposed framework. (a) QA instances are converted into knowledge graphs with query, entity, and agent nodes, with edges defined to capture semantic relations. Query–agent edges are trainable to enable adaptive routing. (b) A type-aware heterogeneous RouterGNN propagates contextual and relational information across the graph. The router then predicts a task-dependent distribution over agents, trained via KL divergence against empirical agent performance. Final answers are obtained as the weight vector of the agents per query.

into a knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The central motivation is that the router must reason not only about the query and the available agents, but also about the contextual entities and their relations, since these often determine which reasoning strategies are most effective. Importantly, such contextual information does not directly reveal the best agent; rather, it shapes the structural cues that guide the router in weighting complementary agents. To this end, the graph is designed to contain query nodes, entity nodes, and agent nodes, connected by edges that reflect lexical, semantic, and trainable routing relations.

Nodes as Representations of Context. To faithfully represent the fine-grained contextual information, we design the following node types. 1) Entity nodes are introduced to preserve the contextual signals. We employ a spaCy pipeline to extract named entities, temporal expressions, and numerical mentions from the context, normalizing them into the set $\mathcal{V}_E$. Each entity node stores its surface form, NER type, and frequency, ensuring that repeated mentions amplify the importance of the corresponding node during message passing. 2) Agent nodes form the set $\mathcal{V}_A$, where each node corresponds to one candidate agent defined by its backbone LLM and prompting strategy (e.g., Chain-of-Thought, or ReAct-style reasoning). Representing agents as nodes embeds them into the same latent space as queries and entities, enabling the router to directly learn compatibility signals. 3) Finally, each input question is represented as a query node $v_q \in \mathcal{V}_Q$, whose embedding is initialized via a contextual encoder. This ensures that semantically similar questions occupy nearby regions in the latent space, providing a natural anchor point for supervision.

Edges as Carriers of Relation Signals. Accurate and diverse edges bind the node families to capture both static semantic structure and dynamic routing preferences. Specifically, we design the following types of edges. 1) Query–entity edges link queries to entities they explicitly mention, grounding the question in its evidence context. 2) Entity–entity edges are established through dependency parsing, where we extract relation triples (h, r, t) from spaCy's parser and record them as connections between entity nodes. This step introduces relational structure into the graph, allowing message passing to propagate semantic information across linked entities. 3) To reflect agent perspectives on context, we further construct agent–entity edges by prompting each agent to identify the most relevant entities it attends to. These edges specify which parts of the context are emphasized by different agents. 4) By contrast, query–agent edges are left as trainable connections. Their role is to carry the routing signal that the model learns to predict: the existence and weight of such edges determine which agents participate in collaboration and how strongly their outputs are weighted. This way, frozen edges capture contextual grounding, and trainable query–agent edges encode adaptive routing.

Edge-to-node Mapping for KG Learning. Since linguistic relations (verbs, dependency types) are open-ended, using them directly as edge types would explode the schema with too many edge

categories, but collapsing them into overly coarse classes would also lose valuable contextual information. To solve this problem, we materialize relations as dedicated nodes $\mathcal{V}_{\text{REL}} \subseteq \mathcal{V}_E$. For each extracted triple (h, r, t), we introduce a relation node r and rewire the connection as $h \xrightarrow{\texttt{inc:src}} r \xrightarrow{\texttt{inc:tgt}} t$. This design allows message passing to explicitly traverse relation semantics, while keeping the edge vocabulary manageable. The original text triples are cached for LLM prompting, ensuring that no semantic detail is lost.

Unified Embedding Space and Graph Definition. As such, all nodes are embedded into the shared textual space to make message passing meaningful. Query nodes use contextual embeddings of the question text; Entity and relation nodes use surface text embeddings augmented with type and frequency features; Agent nodes rely on sentence embeddings of their descriptive strategies and prompts. Embedding everything into one space ensures that similarity across node types is preserved before graph-level learning. Although each QA instance yields a unique graph, the agent set is always fixed, thus query–agent edges are consistently trainable. This design ensures that message passing remains effective: structural signals flow through contextual nodes, while routing signals are learned on top of the stable agent-query scaffold. Consequently, we construct the knowledge graph, which provides a principled, semantically grounded representation of QA instances, enabling the router to learn effective collaboration schemes across agents in the next stages.

3.2 Router Training via RouterGNN

With the knowledge graph in place, the router must be trained to determine which agents are most useful for a given downstream task. Unlike prior routers that rely on LLMs-as-judges or heuristic voting rules, our training-based approach allows the router to generalize beyond fixed protocols and adaptively weight agents according to contextual signals. This shift from rule-based selection to supervised learning enables the model to exploit nuanced dependencies between queries, entities, and agents that are inaccessible via static ensembling.

To achieve this, we adopt **RouterGNN**, a heterogeneous Graph Neural Network (GNN) that performs type-aware message passing across the knowledge graph. Each node embedding is first projected into a shared latent space with a type-specific projection operator $\mathrm{Proj}_{\tau(v)}$. For an edge $(u \xrightarrow{\psi} v)$ of type ψ, the message is computed as

$$m_{u\to v}^{(l,\psi)} = \mathrm{Proj}.\Big(W_{\psi}^{(l)}\, h_u^{(l-1)}\Big),$$

and aggregated by mean pooling over all neighbors of type ψ. Messages from different edge types are then combined with learnable gates, producing the node update

$$h_v^{(l)} = U_{\tau(v)}^{(l)}\Big(h_v^{(l-1)} \,\|\, \sum_{\psi\in\Psi(v)} w_{\psi}^{(l)} \cdot \tilde{m}_v^{(l,\psi)}\Big),$$

where $\tau(v)$ is the node type, $w_{\psi}^{(l)}$ is a learned scalar per edge type, $U_{\tau(v)}^{(l)}$ is a type-specific update function, and $\|$ denotes concatenation. This update rule ensures that each node embedding integrates both its previous state and type-aware signals from heterogeneous neighbors.

After L layers, the query embedding $h_q^{(L)}$ encodes contextual and relational evidence, while each agent embedding $h_a^{(L)}$ captures its suitability for the query. Routing scores are then produced by

$$s(q,a) = \mathrm{MLP}\Big(h_q^{(L)} \,\|\, h_a^{(L)}\Big),$$

$$p_\theta(a \mid q, \mathcal{G}) = \mathrm{softmax}_{a\in\mathcal{A}}\big(s(q,a)\big).$$

Supervision is derived from the empirical performance of candidate agents. For each query, we evaluate all agents and transform their F1 scores into a soft target distribution $p^*(a \mid q)$ via a temperature-scaled softmax, yielding smoother and more informative labels than hard one-hot assignments. The router is then trained by minimizing the Kullback–Leibler divergence

$$\mathcal{L}_{\mathrm{KL}}(q) = \sum_{a\in\mathcal{A}} p^*(a \mid q)\, \log \frac{p^*(a \mid q)}{p_\theta(a \mid q, \mathcal{G})}.$$

KL divergence is especially appropriate in this setting. Unlike cross-entropy, which strongly emphasizes only the top-performing label, KL enforces alignment across the entire distribution, ensuring that the router captures the relative strengths of both primary and secondary agents. Compared to mean squared error, KL better respects the geometry of probability distributions and avoids vanishing gradients for low-probability agents. This richer supervision encourages the router not only to identify the best agent but also to approximate the correct balance among complementary agents,

Method	2Wiki		HotpotQA		NewsQA		TriviaQA	
	F1	EM	F1	EM	F1	EM	F1	EM
Average	51.23±1.46	41.87±0.91	59.52±1.33	46.63±0.97	57.96±2.01	36.11±0.84	45.09±2.35	36.62±1.05
Majority Vote	70.12±1.95	61.00±1.00	63.27±0.28	52.80±1.12	59.11±0.62	39.33±1.53	50.16±1.19	39.00±1.00
Best LLM	71.27±1.71	62.19±0.94	68.32±1.65	50.83±1.00	60.42±1.74	40.79±0.63	48.21±1.62	39.12±0.44
Best Agent	74.89±1.12	63.75±1.24	68.68±1.70	56.20±1.06	62.89±2.62	37.14±1.41	59.33±2.05	49.17±0.73
LLM-Blender	66.64±2.24	52.00±2.65	64.97±1.98	50.33±3.06	58.83±0.66	38.00±1.73	48.06±0.93	34.00±1.73
HybridLLM	69.01±0.15	56.00±1.00	59.10±2.16	39.33±2.89	51.96±2.73	28.00±2.16	52.08±1.36	38.67±1.53
GraphRouter	64.09±0.13	56.33±0.58	61.73±0.76	49.67±0.58	64.95±0.33	50.67±0.58	52.37±1.26	42.33±1.53
AgentRouter	74.86±2.34	67.33±0.54	**70.54±0.23**	**57.33±0.58**	65.61±1.35	48.67±1.53	**63.36±0.19**	**51.00±0.88**
K=3	75.39±0.40	68.33±0.58	68.41±1.03	55.33±1.53	**68.22±0.84**	**52.33±0.58**	59.91±1.57	49.67±3.20
K=5	**77.02±0.58**	**70.67±0.57**	69.19±0.56	56.33±0.58	68.13±0.61	51.67±0.58	59.07±0.71	47.67±1.03
K=10	76.35±1.15	70.67±1.15	70.07±0.58	56.67±0.58	66.53±0.93	50.00±1.00	62.67±4.51	50.67±5.86
Oracle	92.95±1.35	87.00±1.73	90.70±1.52	83.67±1.53	81.37±0.17	61.33±1.15	71.78±0.54	60.33±1.53

Table 2: Performance results with baseline methods on the four benchmarks. We report the mean and standard deviation for all results. Best (excluding Oracle) results are in **bold**, second best are underlined.

leading to more stable optimization and more flexible collaboration policies.

At test time, the router produces a distribution $p_\theta(a \mid q, \mathcal{G})$ over agents for each query. Final predictions are generated through weighted voting:

$$\hat{y}(q) = \phi(\{\, y_a(q),\, p_\theta(a \mid q, \mathcal{G}) : a \in \mathcal{A}\}),$$

where $y_a(q)$ is the output of agent a on query q, and ϕ denotes a fusion rule such as weighted majority voting. In this way, the router's learned distribution directly governs how much influence each agent has in the final answer, producing a principled and context-aware collaboration scheme.

4 Experiments

4.1 Experiment Setup

Benchmarks. We conduct experiments on an extensive collection of Question Answering tasks: HotpotQA (Yang et al., 2018) and 2WikiMultihopQA (Ho et al., 2020) for multi-hop complex reasoning, and NewsQA (Trischler et al., 2017) and TriviaQA (Joshi et al., 2017) for factual and direct reasoning. We describe the details of these benchmarks and discuss the split of sets with other details in Appendix-B.1.

Baselines. We consider the following baselines: 1) Simple heuristic ensembling methods, including the best LLM method, average score performance, majority vote method, and the best Agent (always choose the agent among the 24 candidates with the best performance). 2) Classic state-of-the-arts LLM routing baselines, including LLM-Blender (Jiang et al., 2023a), HybridLLM (Ding et al., 2024), and GraphRouter (Feng et al., 2025). 3) Oracle, indicating the theoretical upper bound, where questions are answered by the best agent.

While there are many different designs of agents, we hope to cover as many classic and basic design choices as we can. For our experiment purpose, we choose these six agent designs: Raw (the basic LLM method), Chain-of-Thought (CoT) (Wei et al., 2022), Self-Consistency (SC) (Wang et al., 2023c), React-Reflection (Yao et al., 2023; Shinn et al., 2023), Multi-Agent Debate (MAD) (Du et al., 2024) and Multi-Agent Summary. As demonstrated in Figure 1, different agents perform differently on different LLM backbones. Therefore, we choose four classic LLM backbones of similar scales but from different providers: Llama-3-8b-instruct (MetaAI, 2024); Qwen2.5-7B-Instruct-Turbo (Alibaba, 2025); Mixtral-8x7B-Instruct-v0.1 (MistralAI, 2023); and gpt-oss-20b (OpenAI, 2025). All baseline experiments are run on the same settings and the results are recorded from three consecutive runs to avoid fluctuation. More details of the baselines are in Appendix-B.2.

Evaluation Metrics. Following standard practice on SQuAD (Rajpurkar et al., 2016), we report Exact Match (EM) and F1. EM measures the percentage of predictions that exactly match the gold answer string after normalization (e.g., lowercasing, punctuation). F1 is the token-level harmonic mean of precision and recall between the predicted and gold spans, capturing partial overlap. EM emphasizes strict correctness, while F1 provides a softer measure that rewards partially correct answers.

4.2 Main Results

We present the main results of AgentRouter against strong baselines in Table 2. Across all four benchmarks, our method consistently achieves superior performance, underscoring the strength of our rout-

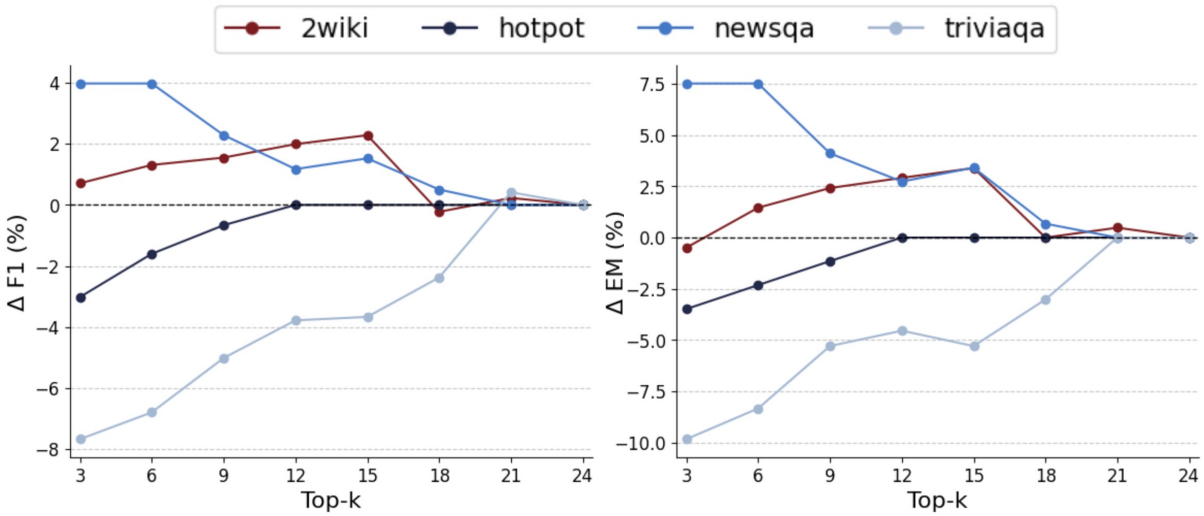

Figure 3: Percentage change (Δ) in F1 (left) and EM (right) relative to $k = 24$, used as the base (0%). Curves show how performance varies with the top k agent clipped across datasets.

ing framework. Several additional insights emerge. First, AgentRouter not only surpasses prior SOTA methods, but also outperforms the best individual agent on each benchmark (Best Agent baseline). This highlights that a collaboration of heterogeneous agents can effectively integrate complementary strengths, yielding higher accuracy than any single agent alone—demonstrating the necessity of adaptive routing. At the same time, the gap to the Oracle remains substantial, suggesting that there is still significant headroom and motivating further exploration of agent routing strategies.

Second, restricting the router's output to the top-K agents (K=3, 5, 10) unexpectedly produces the strongest performance, in some cases exceeding the unrestricted ensemble. A plausible explanation is that pruning the long tail of low-quality or noisy agents reduces variance and sharpens the aggregation of useful reasoning patterns. This suggests that agent routing benefits not only from diversity but also from judicious selection, where a smaller yet high-quality subset provides a better balance between complementarity and noise.

Finally, when comparing across benchmarks, we observe consistent gains on both multi-hop (2Wiki, HotpotQA) and single-hop/direct QA tasks (NewsQA, TriviaQA). The improvements are especially pronounced on multi-hop datasets, where reasoning requires combining evidence from multiple entities, and the advantage of heterogeneous collaboration becomes more salient. By contrast, in direct QA, the gains are smaller but still evident, indicating that even factual retrieval questions benefit from contextualized agent routing. Taken together, these results show that AgentRouter generalizes

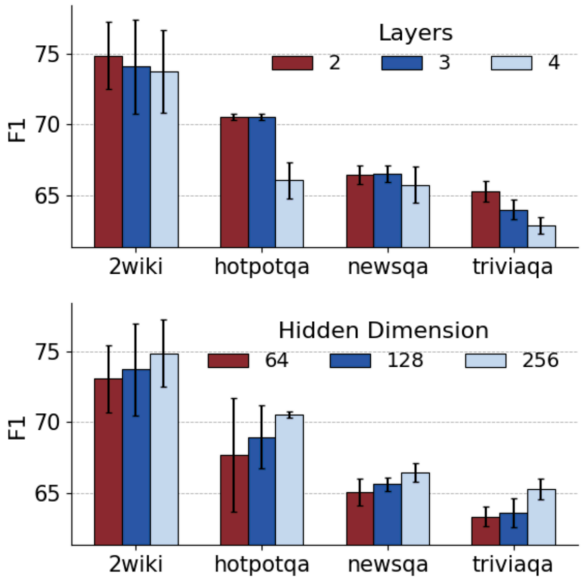

Figure 4: F1 performance on four QA benchmarks, varying Layers (top) and Hidden Dimensions (bottom). Error bars denote standard deviations.

robustly across QA paradigms, with strong benefits in complex multi-hop reasoning scenarios.

4.3 Ablation Analysis

To better understand the effect of restricting the router's output, we conduct an ablation study by varying the number of selected agents K. Figure 3 reports the relative change in F1 and EM, measured against the full ensemble (K=24). Interestingly, we observe that performance does not monotonically increase with larger K. Instead, restricting the router to the top-K agents often yields different results on different benchmarks. Specifically, the clipping method usually works on multi-hop reasoning tasks, where improvements are most pronounced at moderate K (around 5–10), while on

single-hop QA tasks, usually more agents consistently yield better and more stable results. This trend might be due to some agents fail hard on multi-hop reasoning and removing the long tail of weaker agents reduces aggregation noise and variance, allowing more reliable reasoning signals to dominate. Overall, this ablation highlights that agent routing benefits from careful selection rather than unbounded inclusion on multi-hop complex reasoning. A smaller but higher-quality subset of agents provides a more stable and effective collaboration mechanism, underscoring the importance of top-K clipping in practice.

We also study the effect of hyper-parameters. For example, Figure 4 shows the analysis of the effect of hidden dimensions and layers. Across benchmarks, we observe that performance differences between configurations are modest but consistent. Increasing the number of layers does not always improve F1, with deeper settings sometimes yielding lower scores, particularly on multi-hop datasets. In contrast, larger hidden dimensions tend to offer more stable gains across tasks, suggesting that representational capacity plays a more robust role than network depth in this setting. These results motivate a closer examination of architectural trade-offs when designing models for diverse QA benchmarks. Similarly, we also conduct a series of other analyses, including the report of the original single agent behaviors on each task. We include these details in Appendix C.

4.4 Transferability Analysis

As discussed in the previous section, agents built on the same backbone often exhibit wide and non-overlapping performance distributions, and their relative ranking varies substantially across tasks. We now examine this phenomenon from the perspective of *transferability of the learned router weights*. Table 3, 4 report results when the router is trained on one dataset and evaluated on another.

Several observations emerge. First, when the router is trained and tested on different tasks, performance generally deteriorates—particularly under strict Top-K clipping—demonstrating that the router assigns high weights to different subsets of agents depending on the task. This highlights the task-specific nature of agent specialization and reinforces the importance of adaptive routing. Second, the magnitude of degradation decreases as more agents are included, since adding a broader pool of candidates partially compensates for task mismatch. Third, transferability is not uniform across benchmarks: when the router is trained on TriviaQA (single-hop factual QA) and tested on 2Wiki (multi-hop reasoning), performance drops sharply, in some cases falling below simple heuristic ensembles. Conversely, training on 2Wiki and testing on TriviaQA also incurs a gap, but the effect is smaller. This asymmetry suggests that skills learned for multi-hop reasoning might be helpful to factual QA, whereas factual QA training fails to equip the router with strategies for complex reasoning. These results underscore a key insight: AGENTROUTER is most effective when trained with task-relevant information, and it is necessary for an agent router to learn fine-grained, task-aware collaboration patterns. Unlike simple heuristic methods, our framework dynamically adapts to the demands of each benchmark, yielding better results.

Top-K	2Wiki → HotpotQA		2Wiki → TriviaQA	
	F1 Drop%	EM Drop%	F1 Drop%	EM Drop%
3	0.04	-0.61	11.72	10.94
6	0.00	0.00	11.97	14.88
9	-0.24	0.00	13.62	17.61
12	0.43	1.15	14.73	17.45
15	0.58	1.15	11.80	13.61
18	1.66	2.32	11.70	14.06
21	1.98	2.32	14.09	15.91
24	1.98	2.32	11.87	13.64

Table 3: Performance drops when training on 2Wiki.

Top-K	TriviaQA → 2Wiki		TriviaQA → NewsQA	
	F1 Drop%	EM Drop%	F1 Drop%	EM Drop%
3	35.04	41.94	17.53	42.67
6	37.84	47.85	16.99	40.76
9	32.07	40.75	15.08	37.50
12	31.37	39.15	10.52	34.00
15	30.91	38.97	7.39	26.49
18	28.62	36.90	7.76	23.12
21	29.07	38.16	6.66	21.25
24	29.41	38.36	6.08	19.87

Table 4: Performance drops when training on TriviaQA.

5 Conclusion

We introduced AGENTROUTER, a knowledge-graph–guided agent routing framework. By modeling contexts and agents in KG and learning task-aware collaboration strategies, our method consistently surpasses strong baselines across diverse QA benchmarks. Our results demonstrate that explicitly learning contextual information is critical: by guiding routing decisions with contextual and supervised signals, AGENTROUTER is able to capture subtle task-dependent signals that simple heuristics miss, shedding light on new research paths.

Acknowledgements

This work was partially supported by the NSF under grants IIS-2533550, IIS-2321504, IIS-2217239, CNS-2426514, and CMMI-2146076, Notre Dame Strategic Framework Research Grant (2025), and Notre Dame Poverty Research Package (2025). Any expressed opinions, findings, and conclusions or recommendations are those of the authors and do not necessarily reflect the views of the sponsors.

Limitations

While our study provides compelling evidence of the effectiveness of graph-supervised multi-agent routing, two limitations merit discussion. First, we did not explicitly address cost–performance trade-offs. Many routing approaches emphasize efficiency and cost reduction, whereas our design prioritizes performance and robustness. This choice is deliberate: our results show that AGENTROUTER achieves substantially higher accuracy than prior methods, highlighting the value of maximizing performance as a complementary perspective. Nevertheless, future work could incorporate cost-aware objectives to achieve more balanced trade-offs between efficiency and accuracy.

Second, our framework currently operates on a fixed pool of manually designed agents. Although this captures diverse and representative strategies, it inherently limits exploration of the broader design space. Recent advances in automated agent design suggest that infinitely many agent variants could be generated and adapted dynamically. Integrating such automated generation into our routing framework represents a promising direction for future work, enabling continuous expansion of agent diversity. Importantly, our current results indicate that even within a limited agent pool, AGENTROUTER already achieves near-optimal performance relative to strong baselines, providing a solid foundation upon which more flexible and scalable extensions can be built.

References

Qwen Alibaba. 2025. Qwen2.5-7b-instruct. https://huggingface.co/Qwen/Qwen2.5-7B-Instruct.

Yuxuan Cao, Jiarong Xu, Carl Yang, Jiaan Wang, Yunchao Zhang, Chunping Wang, Lei Chen, and Yang Yang. 2023. When to pre-train graph neural networks? from data generation perspective! In *KDD*.

Chaoran Chen, Zhiping Zhang, Bingcan Guo, Shang Ma, Ibrahim Khalilov, Simret Gebreegziabher, Yanfang Ye, Ziang Xiao, Yaxing Yao, Tianshi Li, et al. 2025a. The obvious invisible threat: Llm-powered gui agents' vulnerability to fine-print injections. In *Soups*. The Twenty-First Symposium on Usable Privacy and Security (SOUPS).

Chaoran Chen, Daodao Zhou, Yanfang Ye, Toby Jia-jun Li, and Yaxing Yao. 2025b. Clear: Towards contextual llm-empowered privacy policy analysis and risk generation for large language model applications. In *Proceedings of the 30th International Conference on Intelligent User Interfaces*, pages 277–297.

Justin Chih-Yao Chen, Swarnadeep Saha, and Mohit Bansal. 2024a. Reconcile: Round-table conference improves reasoning via consensus among diverse LLMs. In *ACL*.

Shuhao Chen, Weisen Jiang, Baijiong Lin, James Kwok, and Yu Zhang. 2024b. Routerdc: Query-based router by dual contrastive learning for assembling large language models. *NeurIPS*.

Shuhao Chen, Weisen Jiang, Baijiong Lin, James T. Kwok, and Yu Zhang. 2024c. Routerdc: query-based router by dual contrastive learning for assembling large language models. In *NeurIPS*.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, et al. 2024d. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *ICLR*.

D. Ding, Ankur Mallick, Shaokun Zhang, Chi Wang, et al. 2025. Best-route: Adaptive llm routing with test-time optimal compute. In *ICML*.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks VS Lakshmanan, and Ahmed Hassan Awadallah. 2024. Hybridllm: Cost-efficient and quality-aware query routing. In *ICLR*.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2024. Improving factuality and reasoning in language models through multi-agent debate. In *ICML*.

Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. 2023. Talk like a graph: Encoding graphs for large language models. *arXiv*.

Yihan Feng, Tianyu Zhao, Haotian Liu, Diyi Yang, and Tuo Zhao. 2025. Graphrouter: Learning graph-based routing for large language model selection. In *ICLR*.

Yifu Gao, Linbo Qiao, Zhigang Kan, Zhihua Wen, Yongquan He, and Dongsheng Li. 2024. Two-stage generative question answering on temporal knowledge graph using large language models. *arXiv*.

Siyuan Guo, Cheng Deng, Ying Wen, Hechang Chen, Yi Chang, and Jun Wang. 2024a. Ds-agent: automated data science by empowering large language models with case-based reasoning. In *ICML*.

Tiezheng Guo, Qingwen Yang, Chen Wang, Yanyi Liu, Pan Li, Jiawei Tang, Dapeng Li, and Yingyou Wen. 2024b. Knowledgenavigator: Leveraging large language models for enhanced reasoning over knowledge graph. *Complex & Intelligent Systems*.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *NeurIPS*.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *arXiv*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *COLING*.

Jiatan Huang, Zheyuan Zhang, Tianyi Ma, Mingchen Li, Yaning Zheng, Yanfang Ye, and Chuxu Zhang. 2026a. Glen-bench: A graph-language based benchmark for nutritional health. *arXiv preprint arXiv:2601.18106*.

Jiatan Huang, Zheyuan Zhang, Kaiwen Shi, Yanfang Ye, and Chuxu Zhang. 2026b. Evolverouter: Co-evolving routing and prompt for multi-agent question answering. *arXiv preprint arXiv:2604.05149*.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023a. LLM-blender: Ensembling large language models with pairwise ranking and generative fusion. In *ACL*.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023b. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. In *ACL*.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023c. Structgpt: A general framework for large language model to reason over structured data. In *EMNLP*.

Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, Yang Song, Chen Zhu, Hengshu Zhu, and Ji-Rong Wen. 2024. Kg-agent: An efficient autonomous agent framework for complex reasoning over knowledge graph. *arXiv*.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. 2024. Swe-bench: Can language models resolve real-world github issues? In *ICLR*.

Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Zheng Li, Ruirui Li, Xianfeng Tang, Suhang Wang, Yu Meng, et al. 2024. Graph chain-of-thought: Augmenting large language models by reasoning on graphs. In *ACL*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*.

Mingxuan Ju, Wenhao Yu, Tong Zhao, Chuxu Zhang, and Yanfang Ye. 2022. Grape: Knowledge graph enhanced passage reader for open-domain question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 169–181.

Jiho Kim, Yeonsu Kwon, Yohan Jo, and Edward Choi. 2023. Kg-gpt: A general framework for reasoning on knowledge graphs using large language models. In *EMNLP*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *NeurIPS*.

Angeliki Lazaridou, Elena Gribovskaya, Wojciech Stokowiec, and Nikolai Grigorev. 2022. Internet-augmented language models through few-shot prompting for open-domain question answering. *arXiv*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *NeuralIPS*.

Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Li Erran Li, Ruohan Zhang, et al. 2024a. Embodied agent interface: benchmarking llms for embodied decision making. In *NeuralIPS*.

Peiyu Li, Xiaobao Huang, Ting Hua, and Nitesh V Chawla. 2025a. Crochetbench: Can vision-language models move from describing to doing in crochet domain? *arXiv*.

Peiyu Li, Xiaobao Huang, Yijun Tian, and Nitesh V Chawla. 2024b. Cheffusion: Multimodal foundation model integrating recipe and food image generation. In *CIKM*.

Peiyu Li, Xiuxiu Tang, Si Chen, Ying Cheng, Ronald Metoyer, Ting Hua, and Nitesh V Chawla. 2025b. Adaptive testing for llm evaluation: A psychometric alternative to static benchmarks. *arXiv*.

Yiyang Li, Zehong Wang, Zhengqing Yuan, Zheyuan Zhang, Keerthiram Murugesan, Chuxu Zhang, and Yanfang Ye. 2026. Interpretable graph-language

modeling for detecting youth illicit drug use. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 3630–3647.

Zheyuan Liu, Xiaoxin He, Yijun Tian, and Nitesh V Chawla. 2024. Can we soft prompt llms for graph learning tasks? In *WWW*, pages 481–484.

Tianyi Ma, Yiyang Li, Yiyue Qian, Zheyuan Zhang, Zehong Wang, Chuxu Zhang, and Yanfang Ye. 2026a. Opbench: A graph benchmark to combat the opioid crisis. *arXiv preprint arXiv:2602.14602*.

Tianyi Ma, Yiyue Qian, Zehong Wang, Zheyuan Zhang, Chuxu Zhang, and Yanfang Ye. 2025a. Llm-empowered class imbalanced graph prompt learning for online drug trafficking detection. *arXiv*.

Tianyi Ma, Yiyue Qian, Zehong Wang, Zheyuan Zhang, Chuxu Zhang, and Yanfang Ye. 2026b. Bhygnn+: Unsupervised representation learning for heterophilic hypergraphs. *arXiv preprint arXiv:2602.14919*.

Tianyi Ma, Yiyue Qian, Chuxu Zhang, and Yanfang Ye. 2023. Hypergraph contrastive learning for drug trafficking community detection. In *ICDM*.

Tianyi Ma, Yiyue Qian, Shinan Zhang, Chuxu Zhang, and Yanfang Ye. 2025b. Adaptive expansion for hypergraph learning. *arXiv preprint arXiv:2502.15564*.

Tianyi Ma, Yiyue Qian, Zheyuan Zhang, Zehong Wang, Xiaoye Qian, Feifan Bai, Yifan Ding, Xuwei Luo, Shinan Zhang, Keerthiram Murugesan, et al. 2025c. Autodata: A multi-agent system for open web data collection. *arXiv preprint arXiv:2505.15859*.

MetaAI. 2024. Llama-3-8b-instruct. https://huggingface.co/meta-llama/Llama-3-8B-Instruct.

MistralAI. 2023. Mixtral-8x7b-instruct-v0.1. https://huggingface.co/mistralai/Mixtral-8x7B-Instruct-v0.1.

Bo Ni, Zheyuan Liu, Leyao Wang, Yongjia Lei, Yuying Zhao, Xueqi Cheng, Qingkai Zeng, Luna Dong, Yinglong Xia, Krishnaram Kenthapadi, et al. 2025. Towards trustworthy retrieval augmented generation for large language models: A survey. *arXiv*.

Stefan Nielsen, Edoardo Cetin, Peter Schwendeman, Qi Sun, Jinglue Xu, and Yujin Tang. 2026. Learning to orchestrate agents in natural language with the conductor. In *arXiv*.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. 2025a. Routellm: Learning to route llms with preference data. In *ICLR*.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. 2025b. Routellm: Learning to route llms with preference data. In *ICLR*.

OpenAI. 2025. Gpt-oss-20b. https://llm-stats.com/models/gpt-oss-20b.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *NeuralIPS*.

Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. 2024. Graph retrieval-augmented generation: A survey. *arXiv*.

Yiyue Qian, Tianyi Ma, Chuxu Zhang, and Yanfang Ye. 2024. Dual-level hypergraph contrastive learning with adaptive temperature enhancement. In *WWW*.

Yiyue Qian, Chunhui Zhang, Yiming Zhang, Qianlong Wen, Yanfang Ye, and Chuxu Zhang. 2022. Comodality graph contrastive learning for imbalanced node classification. *Advances in Neural Information Processing Systems*, 35:15862–15874.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Kaiwen Shi, Zheyuan Zhang, Zhengqing Yuan, Keerthiram Murugesan, Vincent Galassi, Chuxu Zhang, and Yanfang Ye. 2026. Ng-router: Graph-supervised multi-agent collaboration for nutrition question answering. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7508–7527.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *NeurIPS*.

Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. 2025. Agentic retrieval-augmented generation: A survey on agentic rag. *arXiv*.

D. Stripelis et al. 2024. A multi-model router for efficient llm inference. In *EMNLP*.

Haitian Sun, Tania Bedrax-Weiss, and William Cohen. 2019. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. In *EMNLP-IJCNLP*.

Dhaval Taunk, Lakshya Khanna, Siri Venkata Pavan Kumar Kandru, Vasudeva Varma, Charu Sharma, and Makarand Tapaswi. 2023. Grapeqa: Graph augmentation and pruning to enhance question-answering. In *WWW*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. Newsqa: A machine comprehension dataset. In *RepL4NLP*.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv*.

Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2024a. Can language models solve graph problems in natural language? *NeuralIPS*.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. 2025a. Mixture-of-agents enhances large language model capabilities. In *ICLR*.

X. Wang, Y. Fu, Y. Zhang, W. Cheng, et al. 2025b. Mixllm: Dynamic routing in mixed large language models. In *NAACL*.

Xintao Wang, Qianwen Yang, Yongting Qiu, Jiaqing Liang, Qianyu He, Zhouhong Gu, Yanghua Xiao, and Wei Wang. 2023a. Knowledgpt: Enhancing large language models with retrieval and storage access on knowledge bases. *arXiv*.

Xinyuan Wang, Yanchi Liu, Wei Cheng, Xujiang Zhao, Zhengzhang Chen, Wenchao Yu, Yanjie Fu, and Haifeng Chen. 2025c. Mixllm: Dynamic routing in mixed large language models. In *NAACL*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *ICLR*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2023c. Self-consistency improves chain of thought reasoning in language models. In *ICLR*.

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

Zehong Wang, Sidney Liu, Zheyuan Zhang, Tianyi Ma, Chuxu Zhang, and Yanfang Ye. 2025d. Can llms convert graphs to text-attributed graphs? In *NAACL*.

Zehong Wang, Zheyuan Zhang, Nitesh Chawla, Chuxu Zhang, and Yanfang Ye. 2024b. Gft: Graph foundation model with transferable tree vocabulary. *NeruIPS*.

Zehong Wang, Zheyuan Zhang, Tianyi Ma, Nitesh V Chawla, Chuxu Zhang, and Yanfang Ye. 2025e. Learning cross-task generalities across graphs via task-trees. *ICML*.

Zehong Wang, Zheyuan Zhang, Tianyi Ma, Nitesh V Chawla, Chuxu Zhang, and Yanfang Ye. 2025f. Neural graph pattern machine. *ICML*.

Zehong Wang, Zheyuan Zhang, Chuxu Zhang, and Yanfang Ye. 2024c. Subgraph pooling: tackling negative transfer on graphs. In *IJCAI*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *NeuralIPS*.

Yilin Wen, Zifeng Wang, and Jimeng Sun. 2023. Mindmap: Knowledge graph prompting sparks graph of thoughts in large language models. *arXiv*.

Shanglin Wu, Yuyang Luo, Yueqing Liang, Kaiwen Shi, Yanfang Ye, Ali Payani, and Kai Shu. 2026. Scaling teams or scaling time? memory enabled lifelong learning in llm multi-agent systems. *arXiv preprint arXiv:2604.03295*.

Siheng Xiong, Oguzhan Gungordu, Blair Johnson, James C Kerce, and Faramarz Fekri. 2026. Scaling search-augmented llm reasoning via adaptive information control. *arXiv preprint arXiv:2602.01672*.

Siheng Xiong, Ali Payani, and Faramarz Fekri. 2025a. Enhancing language model reasoning with structured multi-level modeling. In *The Fourteenth International Conference on Learning Representations*.

Siheng Xiong, Ali Payani, Yu'an Yang, and Faramarz Fekri. 2025b. Deliberate reasoning in language models as structure-aware planning with an accurate world model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 31900–31931.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *ICLR*.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. In *NAACL*.

Yanfang Ye, Zheyuan Zhang, Tianyi Ma, Zehong Wang, Yiyang Li, Shifu Hou, Weixiang Sun, Kaiwen Shi, Yijun Ma, Wei Song, et al. 2025. Llms4all: A review of large language models across academic disciplines. *arXiv preprint arXiv:2509.19580*.

Yanwei Yue, Guibin Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyan Qi. 2025. Masrouter: Learning to route llms for multi-agent systems. In *ACL*.

Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. 2025a. G-designer: Architecting multi-agent communication topologies via graph neural networks. In *ICML*.

Haozhen Zhang, Tao Feng, and Jiaxuan You. 2025b. Router-r1: Teaching llms multi-round routing and aggregation via reinforcement learning. In *NeurIPS*.

Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. 2022. Subgraph retrieval enhanced model for multi-hop knowledge base question answering. In *ACL*.

Zheyuan Zhang, Lin Ge, Hongjiang Li, Weicheng Zhu, Chuxu Zhang, and Yanfang Ye. 2026. Mapro: Recasting multi-agent prompt optimization as maximum a posteriori inference. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 4458–4480.

Zheyuan Zhang, Yiyang Li, Nhi Ha Lan Le, Zehong Wang, Tianyi Ma, Vincent Galassi, Keerthiram Murugesan, Nuno Moniz, Werner Geyer, Nitesh V Chawla, et al. 2025c. Ngqa: A nutritional graph question answering benchmark for personalized health-aware nutritional reasoning. *ACL*.

Zheyuan Zhang, Zehong Wang, Shifu Hou, Evan Hall, Landon Bachman, Jasmine White, Vincent Galassi, Nitesh V Chawla, Chuxu Zhang, and Yanfang Ye. 2024a. Diet-odin: A novel framework for opioid misuse detection with interpretable dietary patterns. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6312–6323.

Zheyuan Zhang, Zehong Wang, Tianyi Ma, Varun Sameer Taneja, Sofia Nelson, Nhi Ha Lan Le, Keerthiram Murugesan, Mingxuan Ju, Nitesh V Chawla, Chuxu Zhang, et al. 2024b. Mopi-hfrs: A multi-objective personalized health-aware food recommendation system with llm-enhanced interpretation. *arXiv*.

Jianan Zhao, Qianlong Wen, Mingxuan Ju, Chuxu Zhang, and Yanfang Ye. 2023. Self-supervised graph structure refinement for graph neural networks. In *Proceedings of the sixteenth ACM international conference on web search and data mining*, pages 159–167.

Jianan Zhao, Qianlong Wen, Shiyu Sun, Yanfang Ye, and Chuxu Zhang. 2021. Multi-view self-supervised heterogeneous graph embedding. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 319–334. Springer.

A Related Works

A.1 Task-Adaptive Agent Selection

Large language models (LLMs) and agentic frameworks have advanced rapidly in recent years (Ye et al., 2025; Xiong et al., 2025b,a; Chen et al., 2025a,b; Xiong et al., 2026; Li et al., 2024b, 2025b,a). With the rise of LLM-based agents, studies increasingly show that no single LLM or agent uniformly excels across downstream tasks; rather, different models or agents exhibit *complementary strengths*. For example, AgentVerse demonstrates that multi-agent groups can outperform individual agents and display emergent collaborative behaviors (Chen et al., 2024d), while ReConcile shows that organizing diverse LLMs into rounds of discussion with consensus voting yields superior reasoning compared to single-model baselines (Chen et al., 2024a). Given these findings, a natural next step is input-conditioned *selection and coordination*—that is, learning to route or assemble specialized models and agents per query to harvest complementary gains. Early efforts pursued this idea through simple binary or ensemble-based collaboration: majority-vote aggregation (e.g., self-consistency (Wang et al., 2022)) and static ensembling confirmed the benefit of diversity but fixed the model set in advance, offering little adaptivity to input characteristics and limited guidance on which agent should be prioritized for a given instance (Jiang et al., 2023a). Subsequent research introduced learned routers that predict which LLM is best suited for a query, thereby providing more adaptive selection than static ensembles. RouteLLM, for instance, learns routing from human preference data and generalizes across strong–weak model pairs (Ong et al., 2025b). RouterDC employs dual contrastive learning to assemble multiple LLMs, achieving gains over top individual models both in-distribution and out-of-distribution (Chen et al., 2024b). MixLLM frames routing as a contextual bandit problem, enabling dynamic adaptation under evolving query distributions and mixed LLM pools (Wang et al., 2025b). In parallel, other advances emphasized coordination among specialized agents rather than selecting a single model. Approaches such as TO-Router and BEST-Route adaptively determine how many outputs to sample or which expert models to involve depending on query difficulty, rather than relying on fixed pipelines (Stripelis et al., 2024; Ding et al., 2025). While these approaches highlight the value

of adaptivity, many still rely on heuristic collaboration rules or shallow controllers, and they seldom capture richer inter-dependencies among queries, tasks, and agents.

The most recent wave of research formulates routing as a structured learning problem. For example, GRAPHROUTER casts routing as a link prediction task on a heterogeneous graph and leverages graph neural networks to jointly model query–model, query–query, and model–model relations (Feng et al., 2025). This represents a clear step beyond shallow controllers or pairwise scoring, yet it remains limited in its ability to incorporate fine-grained task semantics or supervised graph signals that can more directly guide adaptive collaboration across diverse agent designs.

A.2 Knowledge Graph Question Answering

Research on Knowledge Graph Question Answering (KGQA) has progressed from classic semantic parsing and retrieval paradigms to increasingly model-driven solutions. Early systems translated natural-language questions into executable logical forms (e.g., SPARQL) over a knowledge graph (Sun et al., 2019; Zhang et al., 2022), often pairing pre-trained encoders such as BERT with graph-aware architectures (GNNs/LSTMs) to locate entities, relations, and supporting subgraphs (Yasunaga et al., 2021; Taunk et al., 2023). More recent approaches incorporate large language models (LLMs) to improve both access and reasoning: some convert questions into structured queries like SQL/SPARQL to sharpen retrieval (Jiang et al., 2023c; Wang et al., 2023a), while others emphasize multi-hop inference over retrieved triples or subgraphs to handle compositional reasoning (Kim et al., 2023; Gao et al., 2024; Ma et al., 2026a). Despite these advances, widely used benchmarks remain largely general-purpose and do not fully capture domain-specific demands—e.g., the nuanced constraints present in nutritional-health reasoning scenarios.

A.3 Graph-Retrieval Augmented Generation

Graph-Retrieval Augmented Generation (Graph-RAG) generalizes the RAG paradigm (Lewis et al., 2020; Ni et al., 2025) by retrieving *structured* evidence rather than only unstructured text. Instead of passages alone, Graph-RAG surfaces graph fragments (triples/subgraphs) and uses graph encoders to condition generation, thereby improving precision and reducing redundancy (Guo et al., 2024b; Wen et al., 2023; Lazaridou et al., 2022; Liu et al., 2024; Li et al., 2026). Current evaluations predominantly probe elementary graph reasoning skills—such as path finding, degree/counting, or edge existence (Fatemi et al., 2023; Wang et al., 2024a, 2025d). While informative for fundamentals, these settings under-represent domain-specific requirements. He et al. introduce more advanced graph-understanding benchmarks in general contexts (He et al., 2024), yet tailored evaluations for domains like nutrition remain scarce (Zhang et al., 2024b, 2025c; Huang et al., 2026a). Building on Graph-RAG principles, many applications nowadays thrive and shed lights on new research paths (Zhang et al., 2024a)

A.4 Graph Neural Networks.

Graph Neural Networks (GNNs) (Zhao et al., 2023; Qian et al., 2022; Ju et al., 2022; Zhao et al., 2021) are designed for relational data and have delivered strong results across social, recommendation, biological, and molecular applications by exploiting graph inductive biases (Kipf and Welling, 2016; Veličković et al., 2017; Hamilton et al., 2017; Ma et al., 2026b). Their ability to share parameters across varying graph sizes/topologies supports deployment in dynamic, real-world settings. A growing body of work investigates transfer and pretraining for cross-task/domain generalization—mirroring trends in language and vision—via subgraph pooling, pretraining schemes, and task-agnostic embeddings (Ma et al., 2023; Wang et al., 2024c; Ma et al., 2025a; Cao et al., 2023; Ma et al., 2025b). Looking forward, the community is moving toward *graph foundation models*, i.e., large-scale pretrained GNN backbones intended to capture broadly reusable structural/semantic patterns (Wang et al., 2024b; Qian et al., 2024; Wang et al., 2025f,e). Despite progress, open challenges persist, including over-smoothing, expressive-power limits, and scalability, motivating research on more adaptive architectures and training recipes.

A.5 LLM Routing and Multi-Agent Collaboration

Combining outputs from multiple LLMs to outperform any single model has become an active area of research (Wu et al., 2026). Early work mainly relied on heuristic aggregation (Huang et al., 2026b). Mixture-of-Agents (Wang et al., 2025a) showed that stacking agents in layers with equal weights can outperform leading models.

Self-Consistency (Wang et al., 2023b) and LLM-Blender (Jiang et al., 2023b) introduced majority voting and pairwise ranking, respectively. However, these methods apply the same aggregation strategy to all queries and therefore lack adaptivity.

This limitation motivated learned routers. RouteLLM (Ong et al., 2025a), RouterDC (Chen et al., 2024c), and MixLLM (Wang et al., 2025c) learn query-level routing decisions through preference learning, contrastive learning, and contextual bandits. More structure-aware methods further model collaboration explicitly. MasRouter (Yue et al., 2025) uses a cascaded controller to determine collaboration mode, role assignment, and model selection. GraphRouter (Feng et al., 2025) formulates routing as link prediction on a heterogeneous graph, while G-Designer (Zhang et al., 2025a) dynamically generates communication topologies for each task. Another line of work treats the router itself as an LLM. Router-R1 (Zhang et al., 2025b) and Nielsen et al. (Nielsen et al., 2026) train LLM-based orchestrators with reinforcement learning, but these methods incur substantial inference overhead because each routing decision requires a full LLM forward pass.

B Implementation Details

B.1 Benchmarks

HotpotQA (Yang et al., 2018) is a large-scale question answering benchmark explicitly designed to evaluate multi-hop reasoning. Unlike traditional QA datasets where answers can be located within a single passage, HotpotQA requires systems to combine information across multiple documents to arrive at the correct answer. It contains over 100,000 question–answer pairs derived from Wikipedia, annotated with supporting sentences that enable explainable reasoning. This benchmark challenges models not only to retrieve the right evidence but also to demonstrate the ability to integrate scattered information coherently. Given its emphasis on multi-step inference, HotpotQA serves as a critical testbed for evaluating reasoning beyond simple fact extraction.

2WikiMultihopQA (Ho et al., 2020) extends the idea of multi-hop reasoning by constructing questions that require reasoning across pairs of Wikipedia articles. Compared to HotpotQA, it provides more challenging scenarios where supporting evidence is deliberately spread across two distinct documents, forcing models to bridge semantic gaps between disparate sources. The dataset includes a diverse range of question types, from entity relations to compositional reasoning, thereby testing both retrieval and reasoning capabilities. By requiring models to locate and combine information from multiple, sometimes loosely connected passages, 2WikiMultihopQA offers a stringent benchmark for assessing deeper reasoning and robust evidence integration.

NewsQA (Trischler et al., 2017) is a large-scale question answering dataset constructed from CNN news articles. It consists of over 100,000 human-generated questions paired with answers derived from corresponding news passages. Unlike earlier QA datasets that focus on simple fact extraction, NewsQA emphasizes reasoning, inference, and synthesis across multiple sentences within an article. Its design introduces ambiguity, unanswerable questions, and multi-sentence reasoning, making it a challenging benchmark for evaluating reading comprehension and open-domain question answering systems.

TriviaQA (Joshi et al., 2017) is a QA benchmark built from trivia-style questions collected from online sources. It contains over 95,000 question–answer pairs, with evidence passages drawn from Wikipedia and the broader web. TriviaQA is particularly challenging because its questions are authored independently of the supporting documents, resulting in diverse phrasing, lexical variation, and indirect evidence. This property forces models to rely on semantic understanding rather than surface-level matching. As such, TriviaQA tests both retrieval robustness and reasoning generalization under noisy, real-world conditions.

B.2 Baselines

Agent Designs. To capture the breadth of agentic strategies in contemporary LLM research, we incorporate six representative agent designs. *Raw* serves as the direct prompting baseline, providing a view of the backbone's unaugmented capability. *Chain-of-Thought (CoT)* (Wei et al., 2022) elicits intermediate reasoning steps, exposing latent deductive processes that enhance multi-step problem solving. *Self-Consistency (SC)* (Wang et al., 2023c) improves over CoT by sampling multiple reasoning paths and aggregating the most consistent outcome, thereby mitigating variance from any single trajectory. *ReAct-Reflection* (Yao et al., 2023; Shinn et al., 2023) augments reasoning with external actions and iterative self-correction, producing more

Benchmark	Agent	Raw	CoT	SC	React Reflect	MAD	Summary
NewsQA	Qwen2.5-7B	60.42±1.74	59.83±2.63	58.91±1.62	59.95±1.75	59.94±2.81	60.27±1.50
	gpt-oss-20B	59.89±2.62	61.07±1.51	59.36±2.50	59.55±1.64	61.78±1.89	61.69±2.33
	Mixtral-8x7B	54.12±1.35	53.75±2.37	56.10±1.67	54.68±1.46	54.96±2.75	55.51±2.84
	Llama-3-8B	55.24±1.99	56.42±1.25	58.26±2.45	56.82±2.24	56.76±1.86	55.73±1.57
HotpotQA	Qwen2.5-7B	59.69±1.21	57.73±1.32	58.32±1.37	57.20±1.60	55.89±2.48	56.93±2.72
	gpt-oss-20B	68.32±1.65	68.68±1.70	67.35±1.81	68.62±1.04	67.43±1.93	66.40±1.09
	Mixtral-8x7B	58.63±0.47	57.23±0.58	53.49±0.70	58.45±0.86	58.57±0.75	58.75±0.98
	Llama-3-8B	56.50±1.02	54.95±1.14	56.06±1.26	56.31±1.42	54.74±1.37	52.32±1.53
2Wiki	Qwen2.5-7B	41.94±2.20	41.91±1.29	40.92±2.37	44.65±1.45	42.17±1.54	44.56±1.62
	gpt-oss-20B	71.27±1.71	70.63±1.79	71.28±1.87	73.80±1.96	70.50±2.04	74.89±1.12
	Mixtral-8x7B	45.88±0.71	45.12±0.79	43.74±0.87	47.15±0.96	44.30±1.04	47.72±1.12
	Llama-3-8B	48.23±1.21	45.41±1.29	40.85±1.37	41.00±1.46	44.66±1.54	47.04±1.62
TriviaQA	Qwen2.5-7B	35.44±2.67	31.71±1.71	39.00±3.18	35.08±2.10	33.79±2.62	33.99±3.00
	gpt-oss-20B	39.49±2.09	40.85±2.80	44.86±1.74	44.22±2.26	41.23±3.19	39.06±1.88
	Mixtral-8x7B	48.21±1.62	58.42±1.78	59.33±2.05	58.64±2.33	59.21±2.41	56.09±2.74
	Llama-3-8B	46.11±1.76	49.57±2.58	48.07±1.92	46.28±2.04	45.66±2.93	47.90±3.10

Table 5: F1 scores across benchmarks for different agents under various prompting strategies. We report mean and standard deviation.

grounded and robust responses. *Multi-Agent Debate (MAD)* (Du et al., 2024) introduces interactive deliberation among multiple agents, enabling consensus formation through adversarial discussion. Finally, *Multi-Agent Summary* represents collaborative generation where diverse agents contribute partial reasoning that is distilled into a unified answer. These designs span from single-agent prompting to multi-agent coordination, and their diversity is essential for motivating the need for principled routing across heterogeneous strategies.

Simple Heuristic Ensembling. Based on the agent designs, for baselines we first consider straightforward heuristic ensembling methods. While algorithmically simple, these baselines are crucial because they reveal what can be achieved without sophisticated routing. The *best LLM method* reflects the strongest single backbone performance, serving as a natural lower bound for aggregation strategies, as many prior works focus only on LLM routing, overlooking the potential improvement brought by agent designs. *Average score performance* and *majority vote* represent intuitive ways of pooling outputs across agents—either by averaging confidence scores or relying on democratic consensus. Finally, the *best Agent* baseline always selects the single best-performing agent (per task), simulating an optimal and competitive strategy, this is the upper bound of the performance without agent collaboration with each other. Collectively, these heuristics demonstrate how much performance gain can be extracted from raw diversity alone, providing a necessary contrast against more advanced routing frameworks.

Major Routing Baselines. Beyond heuristics, we benchmark against three state-of-the-art LLM routing frameworks. **LLM-Blender** (Jiang et al., 2023a) leverages an LLM-as-a-judge paradigm: candidate outputs from multiple agents are presented to a meta-LLM, which adjudicates and selects the final answer. This approach highlights the potential of reflective meta-reasoning but incurs high cost and latency due to repeated LLM calls. **HybridLLM** (Ding et al., 2024) adopts a hybrid strategy that combines lightweight scoring heuristics with selective meta-LLM adjudication, aiming to balance efficiency and effectiveness. Finally, **GraphRouter** (Feng et al., 2025) formulates routing as a graph-based learning problem, where agents and inputs are embedded into a structured representation, and routing decisions are learned via graph neural networks. This framework demonstrates the benefits of explicitly modeling relational structures among agents and instances, setting a strong precedent for graph-based collaboration learning.

It is worth noting, our work is greatly inspired by the broader idea of graph-based reasoning for LLM orchestration in GraphRouter. However, our work departs fundamentally from GraphRouter in several ways. First, GraphRouter encodes a query and its context as a single node linked to candidate LLMs, effectively reducing the rich internal structure of the context into a coarse representation. In contrast,

Benchmark	Agent	Raw	CoT	SC	React Reflect	MAD	Summary
NewsQA	Qwen2.5-7B	40.79±0.63	39.13±1.34	41.91±0.55	41.88±0.76	41.29±1.57	42.08±0.32
	gpt-oss-20B	37.14±1.41	36.93±0.39	35.22±1.22	36.03±0.47	37.75±0.83	38.17±1.10
	Mixtral-8x7B	28.18±0.43	27.78±1.02	29.27±0.59	29.09±0.36	27.89±1.31	27.24±1.48
	Llama-3-8B	36.84±0.74	38.12±0.28	40.31±1.15	37.06±0.91	38.72±0.67	37.81±0.52
HotpotQA	Qwen2.5-7B	46.83±1.36	46.20±1.42	45.80±1.48	43.86±1.60	43.17±1.54	44.14±1.66
	gpt-oss-20B	50.83±1.00	56.20±1.06	53.80±1.12	54.86±1.24	55.17±1.18	53.14±1.30
	Mixtral-8x7B	44.83±0.28	44.20±0.34	39.80±0.40	44.86±0.52	46.17±0.46	44.14±0.58
	Llama-3-8B	43.83±0.64	42.20±0.70	44.80±0.76	42.86±0.88	42.17±0.82	41.14±0.94
2Wiki	Qwen2.5-7B	32.26±1.30	33.73±1.36	31.29±1.42	35.70±1.48	33.31±1.54	34.68±1.60
	gpt-oss-20B	62.19±0.94	64.80±1.00	61.21±1.06	65.77±1.12	63.24±1.18	63.75±1.24
	Mixtral-8x7B	36.05±0.23	33.94±0.29	34.07±0.35	35.92±0.41	33.09±0.47	36.89±0.53
	Llama-3-8B	38.12±0.59	36.87±0.65	31.14±0.71	32.85±0.77	37.17±0.83	36.82±0.88
TriviaQA	Qwen2.5-7B	29.87±1.41	26.16±0.47	33.22±1.96	28.25±0.85	27.92±1.28	25.91±1.61
	gpt-oss-20B	30.88±0.92	33.19±1.24	36.14±0.57	34.82±1.09	31.71±1.83	31.11±0.62
	Mixtral-8x7B	39.12±0.44	46.88±0.59	49.17±0.73	46.21±1.03	47.09±0.88	42.86±1.17
	Llama-3-8B	39.92±0.51	42.18±1.36	39.11±0.66	38.14±0.79	37.87±1.52	41.23±1.68

Table 6: Exact Match (EM) scores across benchmarks for different agents under various prompting strategies. We report mean and standard deviation.

we decompose queries and contexts into entity-level knowledge graphs, enabling more nuanced modeling of semantic relations that prior research has shown to be critical for reasoning-intensive QA. Second, whereas GraphRouter formulates routing as a naive edge prediction problem between queries and LLMs, our method leverages graph-supervised edge weighting to directly learn collaboration strategies among agents, thereby producing a principled voting scheme rather than a single-model routing decision. Finally, GraphRouter is designed solely for LLM selection, whereas we operate in a multi-agent setting, showing that even when agents share the same backbone, their design choices lead to drastically different behaviors on different tasks. By jointly modeling heterogeneous agents and LLMs, our framework enables emergent collaboration schemes that significantly surpass the strongest single-agent baselines. Also importantly, unlike GraphRouter, our focus is on maximizing accuracy and transferability, rather than cost-effectiveness, offering a complementary perspective in the landscape of graph-based LLM coordination.

B.3 Training Details

Split. We use four datasets—2WikiMultihopQA, HotpotQA, NewsQA, and TriviaQA—with a uniform slicing protocol across splits. For each dataset, we take the *first 500* examples of the official training split as our training data. From the official validation split, we use the *first 100* examples as our validation set and the interval *[100, 200)* (i.e., the next 100 examples) as our test set. This fixed, index-based selection makes experiments easy to replicate across all corpora and avoids leakage between splits while keeping evaluation costs manageable.

Hyperparameters. Unless otherwise specified, we train the RouterGNN with hidden size 256 and 2 layers, optimized by AdamW (learning rate 1×10^{-4}, weight decay 1×10^{-4}) and gradient clipping at 1.0. The device is chosen automatically among `cuda`/`mps`/`cpu` (default `cpu`). Evaluation uses a stable top-k selection over agents with k=24 by default (sorting by $(-p, i)$ to deterministically break ties) and weighted voting for answer aggregation. To construct the soft routing targets, we compute per-agent F1 against gold answers and apply $\mathrm{softmax}(\mathrm{F1}/\tau)$ with τ=0.25 and a small label smoothing ε=10^{-3}. For LLM-backed supervision and logging, we cache agent answers and, when enabled, call the API with temperature 0.2, max tokens 3000.

Miscellaneous. We adopt an early-stopping practice that saves a checkpoint only when the validation F1 strictly improves; ties do not overwrite the earlier best. All reported test numbers come from the validation-best checkpoint. In addition, during graph construction we enrich question nodes with lightweight type cues: a heuristic keyword scan maps tokens such as *which, where, who, when, why, how, whether* to coarse categories (e.g., person, location, time, reason, manner, yes–no), and when dataset-provided question types are available (e.g.,

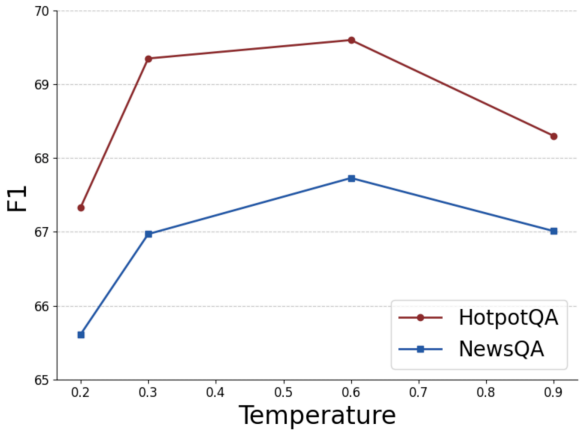

Figure 5: F1 performance on HotpotQA and NewsQA under different temperature settings. Moderate values (0.3–0.6) yield the strongest results.

in 2Wiki/HotpotQA) we merge them as well. These type features serve as a weak prior for the model's type-conditioned routing head without changing supervision targets. Finally, we use Together AI to call the LLM APIs and one NVIDIA GeForce RTX 3090 GPU to train the graph neural networks.

C Additional Experiments

In this section, we report the additional experiments we performed to support our claims in the main sections. First, we report the original agent performance used to calculate the average baseline, the best LLM and the best agent, as can be seen in Table 5 and Table 6. Then, we also report the effect of other hyper-parameters here. Specifically, we also study the effect of LLM temperature as an example. As shown in Figure 5, we observe that moderate settings (0.3–0.6) achieve the best balance across different benchmarks, while overly low or high temperatures lead to degraded performance. This indicates that controlled randomness is beneficial for agent diversity, but excessive variance harms consistency.

D Prompt Design

To demonstrate the exact instructions used in our system, we present the full set of prompts that guided the different agent roles. Figures 6 and 7 provide a complete overview. These prompts are not intended as a novel design contribution, but rather as transparent documentation of the configurations employed in our experiments.

E Case Study

To better understand the effectiveness of our routing mechanism, we conducted a set of qualitative case studies such as those shown in Figure 8. Before discussing the results, it is useful to categorize the typical types of agent errors we observe. These include: (1) *content errors*, where the answer text is semantically incorrect; (2) *format errors*, where the output does not follow the required answer format (e.g., missing a boxed final answer or producing extraneous text); (3) *mixed errors*, where the response contains partially correct information but is polluted with unrelated or contradictory details; and (4) *null responses*, where the agent fails to provide an answer at all (e.g., returning "None" or empty output). These diverse error modes highlight the importance of an intelligent router that can identify which agents are more likely to generate useful and valid responses for a given question.

The three representative examples illustrate how our model's routing module successfully shifts probability mass toward the most reliable agents.

In the graph construction process, we first parse the context into a heterogeneous graph of entities and their dependency or semantic relations. Then, for each **backbone** $\times$ **agent**, we run an **LLM-as-judge** process: given the question, the local entity subgraph, and the agent's prompt or capability profile, the judge scores each (agent, entity) pair and establishes `manage` edges for the highest-scoring pairs — meaning "this agent is responsible for these entities."

In this example (Figure 9), the judge connects llama3_8b::summary to key entities like *The Falcon Takes Over*, and *George Sanders*. The selected agent can then reason step by step through its managed subgraph:

The Falcon Takes Over $\xrightarrow{\texttt{star}}$ *George Sanders* $\xrightarrow{\texttt{prep:as}}$ *Gay Lawrence* $\xrightarrow{\texttt{dep:attr}}$ *the Falcon*.

By following this reasoning chain grounded in the context, the agent is able to reach the correct answer node, therefore receiving the message.

QA/Reasoning Prompt Suite — Part I

debate_debater_a:
You are Debater A. Your goal is to propose the most plausible answer using the provided context.
- Make ONE clear claim (the candidate answer).
- Support it with 1–2 ultra-short quotes (verbatim substrings) and name the hops.
- Explain the link between the quotes in $\leq$2 sentences.

Do NOT use outside knowledge and do NOT output the final boxed answer. Make your answer really short and concise.

debate_debater_b:
You are Debater B. Your goal is to stress-test A's claim using ONLY the provided context.
- If A's quotes or hops are weak, inconsistent, or incomplete, point it out and give corrected quotes/hops.
- If a better candidate exists, state ONE alternative with 1–2 short quotes and $\leq$2 sentences of reasoning.
- If A is already well-supported, briefly confirm but add one missing check.

Do NOT use outside knowledge and do NOT output the final boxed answer. Make your answer really short and concise.

debate_judge:
You are the Judge. Read A and B as supporting analyses and decide the best final answer using ONLY the given context.
If evidence is thin, still make your best context-based guess.
Output MUST include nothing but brief final answer in the format: `\boxed{}`.

react:
You are a multi-hop reasoning expert and an expert QA agent.
Given a question, a news context, and retrieved documents, think step-by-step, silently chain facts to derive a thinking plan,
then use this plan to derive the final brief answer.
Your output format MUST be a brief final answer on the last line in the format: `\boxed{<answer>}`.

reflect:
You are a judge overseeing a multi-hop reasoning expert and an expert QA agent.
Given a question, a news context, and retrieved documents, you will evaluate the agent's answer based on the correctness and notes.
If the answer is incorrect or incomplete, provide constructive feedback and suggest specific revisions to improve the answer.
If the answer is correct and complete, indicate that no further revisions are needed.
Your output MUST end with either:
- "Status: revise" followed by specific feedback and revision suggestions, if the answer needs improvement.
- "Status: final" if the answer is correct and complete.

If you indicate "Status: revise", also include a short "Feedback: <your feedback here>" section before the final answer.

Figure 6: Prompt suite for multi-hop QA (Part I).

QA/Reasoning Prompt Suite — Part II

think_a:
You are a multi-hop reasoning expert and an expert QA agent.
Given a question, a news context, and retrieved documents, think step-by-step, chain facts to derive the answer.
Give your final answer as a single entity, and a concise reasoning process that leads to the answer.

think_b:
You are a multi-hop reasoning expert and an expert QA agent.
Given a question, a news context, and retrieved documents, think step-by-step, chain facts to derive the answer.
Give your final answer as a single entity, and a concise reasoning process that leads to the answer.

summarize:
You are the multi-hop reasoning expert and an expert QA agent. You receive outputs from other agents. Use them as **supporting signals**.
If A and B agree on the same short span, return it. If they differ, pick the best answer with your own reasoning.
Your output format MUST end with the brief final answer on the last line in the format: `\boxed{<answer>}`.

router:
You are a routing model.
Decide if the following user question is CHALLENGING or EASY.
Answer with a single token: `'CHALLENGING'` or `'EASY'`.

note:
Here are the rules you must **STRICTLY** follow:
1. Always return the answer as the SHORTEST exact entity only. The answer is always within 10 words, and usually within 5 words.
2. If the question is yes/no, respond strictly with *yes* or *no* only.
3. For year ranges, never use hyphens; instead, use "from XXXX to YYYY" or "XXXX until YYYY".
4. Do not output sentences, explanations, or phrases with verbs; the answer must be a single entity expression only.
5. One way or another, you must return your best guess, and the final answer must be in the format: `\boxed{<answer>}`.

Figure 7: Prompt suite for multi-hop QA (Part II).

Agent Routing Examples (Three QA Cases)

Example 1
Question: What use British singer Gary?
Gold Answer: iPhone apps
Model Prediction: iPhone apps

Top-4 Agents
`BACKBONE::qwen2p5_7b_turbo::AGENT::raw` → *iPhone apps* (0.062086940)
`BACKBONE::gpt_oss_20b::AGENT::mad` → *iPhone apps* (0.060009051)
`BACKBONE::gpt_oss_20b::AGENT::raw` → *iPhone apps* (0.059776004)
`BACKBONE::gpt_oss_20b::AGENT::react_reflect` → *iPhone apps* (0.059702747)

Last-4 Agents
`BACKBONE::mixtral_8x7b::AGENT::mad` → *iPhone, keyboard, Twitter* (0.003830930)
`BACKBONE::mixtral_8x7b::AGENT::raw` → *iPhone/keyboard* (0.004338742)
`BACKBONE::mixtral_8x7b::AGENT::sc` → *iPhone, keyboard, microphone* (0.004404212)
`BACKBONE::mixtral_8x7b::AGENT::cot` → *iPhone/music* (0.004915467)

Example 2
Question: What caused the accident?
Gold Answer: the bus lost control on a curve
Model Prediction: the bus lost control on a curve

Top-4 Agents
`BACKBONE::qwen2p5_7b_turbo::AGENT::cot` → *the bus lost control on a curve* (0.081045948)
`BACKBONE::qwen2p5_7b_turbo::AGENT::react_reflect` → *the bus lost control on a curve* (0.079336852)
`BACKBONE::qwen2p5_7b_turbo::AGENT::raw` → *the bus lost control on a curve* (0.078211986)
`BACKBONE::qwen2p5_7b_turbo::AGENT::mad` → *the bus lost control on a curve* (0.071492545)

Last-4 Agents
`BACKBONE::mixtral_8x7b::AGENT::mad` → *Bus left highway, rolled over* (0.003636589)
`BACKBONE::mixtral_8x7b::AGENT::summary` → *Bus left highway, rolled over* (0.005194495)
`BACKBONE::mixtral_8x7b::AGENT::sc` → *Lost control on curve* (0.005307447)
`BACKBONE::mixtral_8x7b::AGENT::raw` → *unknown* (0.005436183)

Figure 8: Per-question agent routing snapshots for three examples. Each case shows the question, gold answer, model prediction, and the four highest- and lowest-probability agents with their generated answers.

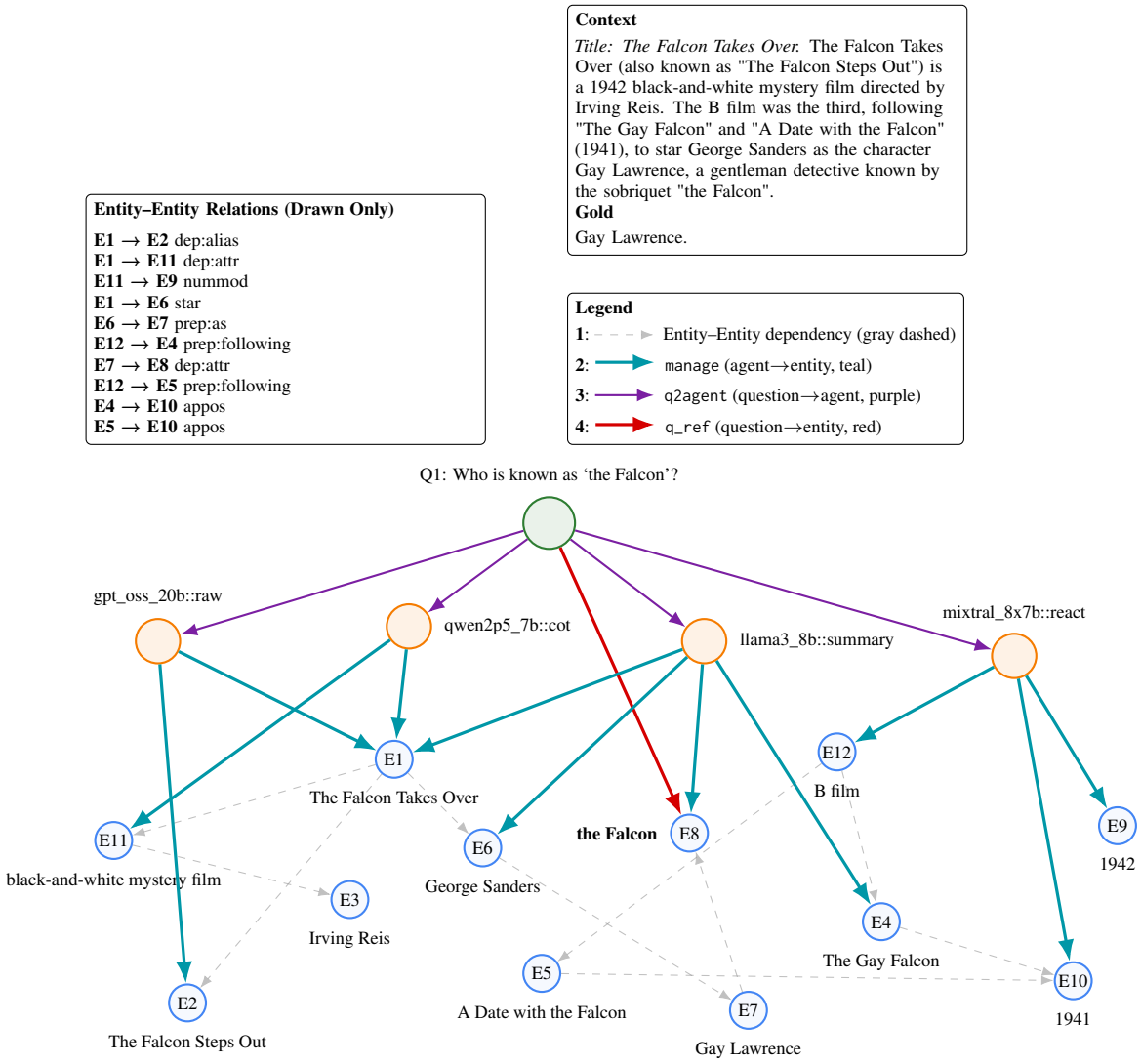

Figure 9: An illustrative example of the heterogeneous graph structure used in our framework. The graph consists of three types of nodes: **question nodes** (green, top), representing input questions; **agent nodes** (orange, middle), corresponding to different reasoning or retrieval agents; and **entity nodes** (blue, bottom), representing extracted knowledge items such as people, films, time expressions, or attributes. Edges capture various types of relations between these nodes: dashed gray edges denote linguistic or semantic relations between entities, teal edges indicate `manage` links connecting agents to the entities they are responsible for, purple edges represent `q2agent` connections from the question to potentially useful agents, and red edges (`q_ref`) link questions directly to referenced entities.