Digest the Knowledge: Large Language Models empowered Message Passing for Knowledge Graph Question Answering

Junhong Wan, Tao Yu, Kunyu Jiang, Yao Fu*, Weihao Jiang, Jiang Zhu
Hikvision Research Institute, Hangzhou, China
{wanjunhong,yutao31,jiangkunyu,fuyao,jiangweihao5,zhujiang.hri}@hikvision.com

Abstract

Despite their success, large language models (LLMs) suffer from notorious hallucination issue. By introducing external knowledge stored in knowledge graphs (KGs), existing methods use paths as the medium to represent the graph information sent into LLMs. However, paths only contain limited graph structure information and are unorganized with redundant sequentially appearing keywords, which are difficult for LLMs to digest. We aim to find a suitable medium that captures the essence of structural knowledge in KGs. Inspired by Neural Message Passing in Graph Neural Networks, we propose Language Message Passing (LMP), which first learns a concise facts graph by iteratively aggregating neighbor entities and transforming them into semantic facts, and then performs Topological Readout that encodes the graph structure information into multi-level lists of texts to augment LLMs. Our method serves as a brand-new innovative framework that brings a new perspective into KG-enhanced LLMs, and also offers human-level semantic explainability with significant performance improvements over existing methods on all five knowledge graph question answering datasets. Our code is available at https://github.com/wanjunhong0/LMP.

1 Introduction

In recent years, large language models (LLMs) have achieved vigorous development and received extensive research attention (Achiam et al., 2023; Touvron et al., 2023). Despite their success, LLMs inevitably suffer from a notorious hallucination issue (Zhang et al., 2023b; Huang et al., 2023), where LLMs inadvertently generate responses that seem plausible but are factually incorrect.

To address this issue, Retrieval-Augmented Generation (RAG) is proposed to enhance LLMs with external knowledge, where augmenting LLMs with more reliable structural knowledge in knowledge graphs (KGs) is a promising solution. Recent works follow a two-phase paradigm that first extracts paths composed of triples from KGs using entity pairs (Wang et al., 2024; Jiang et al., 2023), deep reinforcement learning (Zhang et al., 2023a), and LLMs (Sun et al., 2024; Sui et al., 2024); and directly augments LLMs with the retrieved raw keyword-based triples or paths. Overall, existing methods use paths as the medium to represent the graph knowledge sent into LLMs.

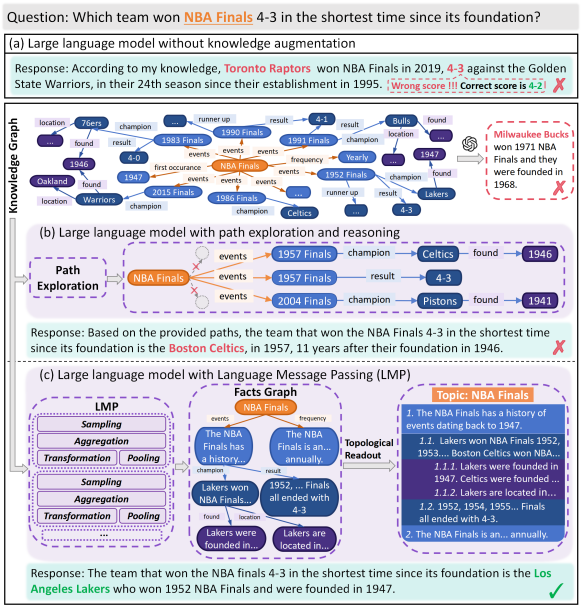

Figure 1: The workflow of vanilla LLM (a) and LLM augmented by paths (b) and facts graph (c).

But are paths a suitable medium to represent the rich graph knowledge for LLMs to digest? Consider the question *which team won the NBA Finals 4-3 in the shortest time since its foundation* that ChatGPT encountered a typical hallucination issue by making up a false score in Figure 1a, which is a superlative question and requires the scores and champion information of all 78 NBA finals to answer. **First, path only explores the**

*Corresponding author.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 15426–15442
July 27 - August 1, 2025

depth information and lacks an overview of the graph structure (Chen et al., 2018). Path-based methods are unable to associate the scores and champions together because there are no direct connections between them and fail to cover the information of all NBA finals needed in Figure 1b. **Second, the raw keyword-based paths are unorganized and semantically awkward for LLMs to digest.** Even with all the necessary information in KGs, LLM still fails at this question in Figure 1 (red dash line box) since the information is unorganized and scattered across separate paths at different depths. Clearly, it is much easier to digest organized information from shallow to deep step by step. Start with there are 78 NBA Finals to consider. Then, find champions with 4-3 score. Finally, provide their foundation time. Additionally, raw paths only contain redundant sequentially appearing keywords, which are semantically awkward for LLMs (Wu et al., 2023).

Since the original KGs are obviously too complex and noisy for LLMs and paths are limited in their ability to serve as the medium, we want to find a suitable medium that captures the essence of structural knowledge in KGs. We draw inspiration from Graph Neural Networks (GNNs), which learn a concise graph that captures high-level features and patterns using Neural Message Passing mechanism (Gilmer et al., 2017) and Readout function (Xu et al., 2019). But directly converting the knowledge stored in neural message passing embedding space for LLMs to understand is difficult. To overcome this, we propose a novel framework called **L**anguage **M**essage **P**assing (LMP), where message passing and final question answering are both performed by LLMs in text space to ensure seamless knowledge transfer. Specifically, as in Figure 1c, we first perform multi-layer language message passing on KG that iteratively filters out irrelevant relations with sampling, aggregates neighbor entities, transforms raw keyword-based entities information into summarized facts and pools the original complex graph into a concise facts graph where each node is a summarized fact. Next, we conduct Topological Readout that performs Depth-First Search (DFS) on the facts graph while assigning the multi-level lists numbering for each node, which encodes both depth and width information of the graph. We augment LLM with the multi-level lists of summarized facts, which are one of the most widely-used structural ways in texts to organize information hierarchically, making it more informative and easier for LLM to digest. The advantages of LMP can be summarized as follows:

- **Brand-new innovative perspective**: To the best of our knowledge, LMP is the first work based on message passing instead of path exploration in existing methods. We incorporate the spirit of GNNs and their benefits into KG-enhanced LLMs, which brings new insight into collaborating LLMs with other well-established methods from different domains for future works.
- **Human-level semantic explainability**: Unlike existing methods that only offer keyword-based paths to interpret the reasoning process, the facts graph in LMP provides semantic facts that contain the entire process of LLM digesting graph information from shallow to deep progressively, offering step-wise human-level explainability.
- **Effective and robust framework**: We validate the effectiveness of LMP on five knowledge graph question answering (KGQA) benchmarks, where LMP achieves SOTA results over existing methods. Additionally, LMP demonstrates significant robustness to noisy KGs, which are frequently encountered in real-world scenarios.

2 Related Works

2.1 Graph Neural Networks

Graph Neural Networks have become the criterion in graph representation learning. The message passing of GNNs includes retrieval, sampling, aggregation, transformation, pooling, and classification. Initially, k-hop ego-net of the target node is retrieved and node sampling is subsequently conducted to reduce computational costs (Hamilton et al., 2017). After that, each node aggregates neighborhood information and updates node representation using neural transformation (Kipf and Welling, 2017; Veličković et al., 2018). Then, pooling combines highly relevant nodes into a hyper-node (Ying et al., 2018). After several iterations, the final representation is used for classification.

2.2 KG-enhanced LLMs

Early studies of KG-enhanced LLMs encode the structural knowledge in KGs as embeddings and incorporate it with the underlying neural networks of LLMs during pre-training (Yu et al., 2022) and fine-tuning(Zhang et al., 2022; Sun et al., 2022). However, KG embedded in LLM requires high computational costs for training, is limited with smaller and less capable LLMs, and also sacrifices the nat-

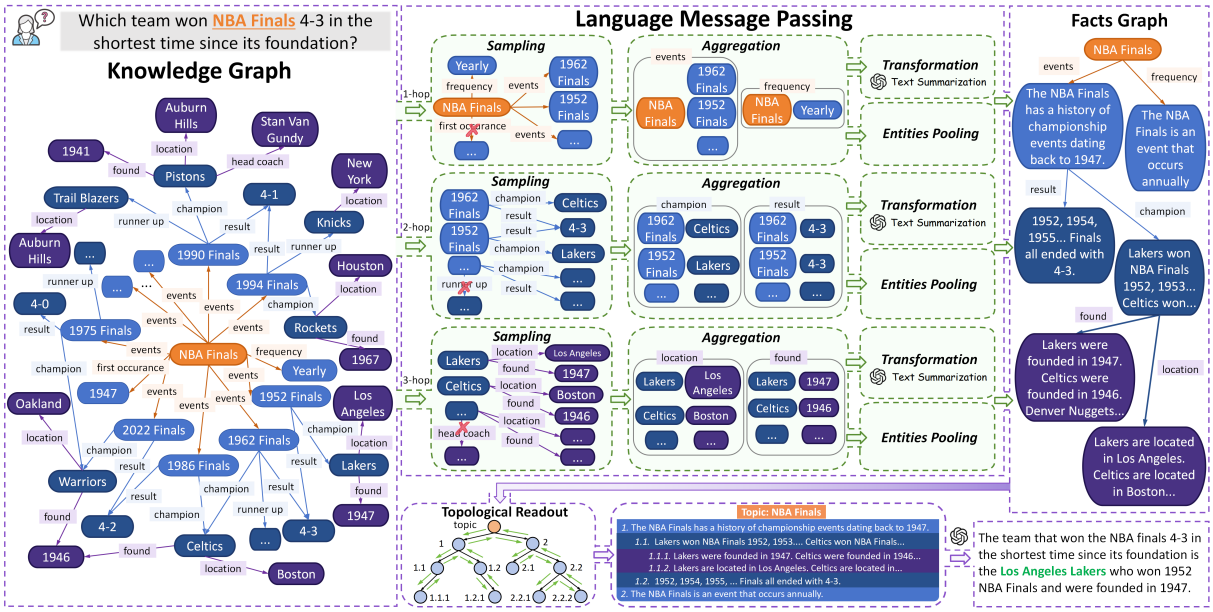

Figure 2: LMP first condenses the original noisy and complex KG (left part) into the facts graph (right part), which iteratively samples the relevant relations of the topic entity (highlighted in orange) in the question, aggregates neighbor entities connecting with those relations, and transforms the aggregated neighborhood information into semantic facts using LLM while pooling the aggregated entities into hyper-entities to form the facts graph. At last, we conduct Topological Readout that performs DFS on the facts graph from the topic entity as the root node while assigning the multi-level lists numbering for each node (bottom part) and then augment LLM to answer the question.

ural explainability in KGs (Sun et al., 2024). To address those limitations, recent works retrieve and translate structural knowledge from KGs to textual prompts for augmenting LLMs. To retrieve knowledge from KG, most methods extract paths composed of triples from KG. RoK (Wang et al., 2024) construct paths based on entity pairs, KnowGPT (Zhang et al., 2023a) uses deep reinforcement learning, ToG (Sun et al., 2024) and RoG (Luo et al., 2024) employ LLMs to extract paths. Although some works conduct BFS-like approaches on KG (Wen et al., 2024), the knowledge retrieved still is presented in collection of paths. As for feeding knowledge into LLMs, all these methods augment LLMs with raw triples or paths. In our work, we incorporate the message passing mechanism and learn a facts graph to better represent the structural knowledge in KGs.

3 Methods

Our task is knowledge graph question answering (KGQA) based on a natural language question q and knowledge graph $\mathcal{G}$. We define the knowledge graph as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{X})$ where $\mathcal{E}$, $\mathcal{R}$, and $\mathcal{X}$ represent the entity set, relation set, and text set, respectively. The text set $\mathcal{X}$ is the text descriptions (e.g. entity names) of entity set $\mathcal{E}$. The topic entity e_q [1] of question q can be obtained by entity linking techniques (Baek et al., 2023; Sun et al., 2024).

3.1 Language Message Passing

Neural Message Passing iteratively aggregates neighbor nodes and transforms their representations by projecting them into new representation spaces through neural networks(Xu et al., 2019), which allows GNNs to learn increasingly abstract and informative representations of the graph structure and node features. To ensure seamless knowledge transfer from graph to text for final question answering, we propose Language Message Passing that leverages the knowledge generalization and processing abilities of LLMs to empower Message Passing instead of neural networks. However, directly performing message passing in GNN's fashion that all nodes are updated in each layer is very computationally expensive for LLMs. We implement a simplified scheme that in l-layer language message passing is only performed on the l-hop relations and neighborhoods of the topic entity, where each node/entity is only updated once during multi-layer language message passing. Unlike the nodes

[1] We only showcase the question with one topic entity for simplicity. For question with multiple topic entities, we simply perform multi-layer language message passing and Topological Readout for each one of them.

in each layer of GNN are in different representation spaces projected by different neural networks, the nodes in our approach are in a common shared text space across layers, where LMP does not need to align the representation space in each layer.

As in Figure 2, LMP performs multi-layer language message passing on KG that iteratively: 1). samples top-K question-related l-th hop relations of the topic entity; 2). aggregates the neighborhood information for each selected relation; 3). transforms aggregated graph information into semantic summarized facts and pools the aggregated entities into hyper-entities. After multi-layer language message passing, we get a facts graph where each node is a summarized fact. The detailed algorithm of LMP is shown in Appendix A.1.

3.1.1 Sampling

Since the scale of KGs like Freebase is enormous and the neighborhoods grow exponentially with the depths increased, performing multi-layer language message passing directly on the question-related subgraph could be extremely time-consuming. Moreover, not every relation is relevant to the question, so augmenting LLM with irrelevant information could bring noise and thus hinder its performance. To address these issues, for l-th layer of language message passing we first perform sampling on the l-hop relations $R^{(l)} \in \mathcal{R}$ of the topic entity to select top K most relevant relations based on the question q and summarized facts from the previous layer $H^{(l-1)}$ as:

$$\hat{R}^{(l)} = \mathrm{TopK}(R^{(l)} \mid H^{(l-1)}, q), \tag{1}$$

where $H^{(0)}$ is initialized as an empty set. We use LLM to perform the TopK operation for better performance and the prompt is in Appendix A.3. This operation can also be implemented using lightweight models (Sun et al., 2024) that measure text similarity such as BM25 or SentenceBERT.

3.1.2 Aggregation

After sampling, we obtain K l-hop relations $\hat{R}^{(l)}$ of topic entity for l-th layer of language message passing. To gain an overview of the l-hop neighborhood information, we aggregate the connecting entities of each selected relation $\hat{r}_k^{(l)} \in \hat{R}^{(l)}$:

$$a_k^{(l)} = \mathrm{Aggregate}(X^{(l-1)}, \hat{r}_k^{(l)}, X_k^{(l)}), \tag{2}$$

where $X^{(l-1)}$ and $X_k^{(l)}$ denote the texts of $l-1$ and l-hop neighbor entities of topic entity with $\hat{r}_k^{(l)}$ connecting between them. $X^{(0)}$ is the text description of the topic entity. The $\mathrm{Aggregate}$ operation is a predefined prompt without LLM involvement detailed in Appendix A.4.

Our aggregation offers two advantages over path-based methods. First, our aggregation prompt is token-efficient in describing neighborhood information and use up to $2/3$ fewer tokens than triples with redundant head entities and relations. Second, and most importantly, the aggregation operation serves as the cornerstone of the robustness of LMP in addressing the challenges posed by incomplete and noisy KGs in real-world scenarios such as missing relations and entities. By aggregating neighborhood information, LMP gains an overview of the local structure of graph, thereby minimizing the impact of missing entities and relations.

3.1.3 Transformation

Since the raw aggregated information is still too lengthy and noisy containing only keywords of relation and entity names, we want to make it concise and further translate it into a format that LLM can better digest. Therefore, we transform the aggregated neighborhood information $a_k^{(l)}$ for each selected relation to summarize while retaining only question-related information:

$$h_k^{(l)} = \mathrm{Transform}(a_k^{(l)} \mid H^{(l-1)}, q), \tag{3}$$

where $H^{(l-1)}$ is summarized facts from the previous layer. We use LLM to conduct $\mathrm{Transform}$ operation and the prompt is in Appendix A.5.

Unlike existing methods that augment LLM with the raw paths, we rarely see deep learning methods that send the raw features directly into downstream tasks without multiple transformation operations to learn the hierarchical and complex relationships within them. With our multiple $\mathrm{Transform}$ operations, LLM gains the understanding of 1-hop graph information before moving to the next-hop, which progressively digests the graph information, from shallow to deep and from easy to hard.

3.1.4 Pooling

Along with transformation that summarizes aggregated neighborhood information $a_k^{(l)}$ for each selected relation, we pool those neighbor entities involved in aggregation into one hyper-entity and assign the summarized fact $h_k^{(l)}$ as its text description:

$$\hat{e}_k^{(l)} = \mathrm{Pool}(E_k^{(l)}), \tag{4}$$

where $E_k^{(l)}$ is the l-hop neighbor entities of the topic entity. The relations connected to pooled entities above are linked to the hyper-entity $\hat{e}_k^{(l)}$.

Since the question-related subgraph may still be too complex with hundreds or even thousands of nodes, we can safely reduce the size of the graph by pooling the l-hop neighbor entities of the topic entity for each selected relation into a hyper-entity where the information from their text descriptions is transformed into summarized fact.

3.2 Topological Readout

After L layers language message passing, we get a facts graph $\hat{\mathcal{G}} = (\hat{\mathcal{E}}, \hat{\mathcal{R}}, \mathcal{H})$ where $\hat{\mathcal{E}}$, $\hat{\mathcal{R}}$, and $\mathcal{H}$ are the pooled hyper-entity set, question-related relation set, and summarized facts serving as the text description set of $\hat{\mathcal{E}}$. To preserve the graph structure information, we perform Topological Readout from the facts graph and augment LLM for the question answering:

$$\text{answer}_q = \text{LLM}(\text{Readout}(\hat{\mathcal{G}}), q). \quad (5)$$

The Readout operation performs DFS on the facts graph from the topic entity as the root node and assigns the multi-level lists numbering for each node in Figure 2. So the facts graph is translated into a multi-level lists of summarized facts, where the list numbers like 1. and 1.1 represent the depth and 1.1 and 1.2 represent the width information of the graph. The multi-level list is one of the most widely used structural ways in texts to organize information hierarchically, which is more informative and easier for LLM to digest. The detailed algorithm is presented in Appendix A.2. At last, we augment LLM with multi-level lists of summarized facts to answer the question. The question answering prompt is in Appendix A.6.

Unlike only augmenting LLM with the raw paths to answer the question, our approach not only preserves the graph structure information but also the step-wise process of LLM searching and digesting information in KG from shallow to deep progressively. This helps LLM to break down a complex question and solve it step by step. Moreover, our facts graph offers explainability with human-level semantic facts to track the reasoning process of LLM, whereas existing methods only provide paths composed of keywords to interpret their results.

4 Experiments

In this section, we aim to answer questions as follows. **Q1:** How does LMP perform against existing methods? **Q2:** How does LMP perform with different LLMs? **Q3:** How does the noise in KG affect LMP? **Q4:** How efficient is LMP compared with others? **Q5:** How do components and hyperparameters impact LMP? **Q6:** How does LMP perform in different types and depths of the question? **Q7:** What explainability does LMP provide?

4.1 Experiment Setups

4.1.1 Datasets & Knowledge Graph

We evaluate LMP on five KGQA datasets: WebQSP (Yih et al., 2016), CWQ (Talmor and Berant, 2018), and GrailQA (Gu et al., 2021) for multi-hop QA; Simple Questions (Bordes et al., 2015) for single-hop QA; WebQuestions (Berant et al., 2013) for open-domain QA. For all datasets, we follow the settings from previous work (Sun et al., 2024). We use Freebase as KG on all datasets (Bollacker et al., 2008), which contains over 120 million entities and 20 thousand relations.

4.1.2 Baselines

We compare our method with three types of baselines, namely LLMs only, fine-tuning, and prompting. The descriptions of baselines are in Appendix B. For all baselines, we report the best results with the most powerful LLM in their original papers.

4.1.3 Implementation Details

LMP is a plug-and-play framework compatible with any open-source or closed API LLMs. We use Llama-2-70B, Llama-3-70B, ChatGPT, and GPT-4 as backbone LLMs, where version of LLMs are detailed in Appendix D.1. The temperature for all LLMs is set to 0. LMP has two main hyperparameters: width K controlling the number of relations kept in each layer, and depth L determining the number of layers of language message passing. Detailed experiment settings in Appendix D.2. The constraint enforcement mechanism with retry is implemented for our sampling and transformation operations to ensure that LLMs produce the desired outputs, as detailed in Appendix A.7. For all datasets, exact match accuracy (Hits@1) is used as the evaluation metric following previous works (Sun et al., 2024; Xu et al., 2024) with F1 score in Appendix E.1.

4.2 Effectiveness Analysis (Q1 & Q2)

For Q1, the main results are shown in Table 1. Clearly, LMP with GPT-4 achieves the best results on all datasets with a significant margin over all

Models		Multi-hop			Single-hop	Open-domain
		WebQSP	CWQ	GrailQA	Simple Questions	WebQuestions
without knowledge graph						
LLMs only	IO + ChatGPT (Brown et al., 2020)	63.3	37.6	29.4	20.0	48.7
	CoT + ChatGPT (Wei et al., 2022)	62.2	38.8	28.1	20.3	48.5
	SC + ChatGPT (Wang et al., 2023)	61.1	45.4	29.6	18.9	50.3
with knowledge graph						
Fine-tuning	DECAF (Yu et al., 2023)	82.1	70.4	-	-	-
	RoG (Luo et al., 2024)	85.7	62.6	-	-	-
	KG-Agent (Jiang et al., 2024)	83.3	72.2	-	-	-
	RRA (Wu et al., 2023)	79.4	-	-	-	73.7
Prompting	ToG-R + GPT-4 (Sun et al., 2024)	81.9	72.5	80.3	58.6	57.1
	ToG + GPT-4 (Sun et al., 2024)	82.6	67.6	81.4	66.7	57.9
	EffiQA + GPT-4 (Dong et al., 2025)	82.9	69.5	78.4	76.5	-
	FiDeLis + GPT-4 (Sui et al., 2024)	84.4	71.5	-	-	-
	KG-CoT + GPT-4 (Zhao et al., 2024)	84.9	62.3	-	86.1	68.0
	LMP + Llama-2-70B	84.3	59.6	79.5	74.4	76.5
	LMP + Llama-3-70B	89.6	72.5	80.6	80.0	76.6
	LMP + ChatGPT	87.2	72.6	82.5	79.2	77.9
	LMP + GPT-4	**90.0**	**82.2**	**89.3**	**86.7**	**80.4**

Table 1: The exact match accuracy (%) of different models in 5 benchmark datasets over 3 distinct domains. The boldface indicates the best performance in the overall results and the underline indicates the second best.

Models	Llama-2-70B		Llama-3-8B		Llama-3-70B		ChatGPT		GPT-4		o1-mini	
	WebQSP	CWQ	WebQSP	CWQ	WebQSP	CWQ	WebQSP	CWQ	WebQSP	CWQ	WebQSP	CWQ
without knowledge graph												
CoT	57.4	39.1	67.7	36.8	71.4	43.0	62.2	38.8	67.3	46.0	75.2	54.6
LMP Gain	*+46.9%*	*+52.4%*	+31.6%	+77.4%	*+25.5%*	*+68.6%*	*+40.2%*	*+87.1%*	*+33.7%*	*+78.7%*	+21.8%	+53.5%
with knowledge graph												
ToG	63.7	53.6	64.4	53.2	-	-	76.2	58.9	82.6	72.5	-	-
EffiQA	-	-	58.3	37.4	-	-	65.2	52.1	82.9	69.5	-	-
LMP Gain	*+32.3%*	*+11.2%*	*+38.4%*	*+22.7%*	-	-	*+14.4%*	*+23.3%*	*+8.6%*	*+13.4%*	-	-
LMP	**84.3**	**59.6**	**89.1**	**65.3**	**89.6**	**72.5**	**87.2**	**72.6**	**90.0**	**82.2**	**91.6**	**83.8**

Table 2: The exact match accuracy (%) of different models under different LLM backbones. The gain percentage with external knowledge is calculated with the best (boldface) and the second best (underlined).

other baselines. Moreover, the second best results are mostly achieved by LMP with less capable LLMs, showcasing the superiority of our methods. Without KG, LLMs only methods obtain the worst results due to the lack of domain-specific knowledge. Compared with fine-tuning methods that have natural advantages from training over prompting methods, LMP still has significant performance improvement. Specifically, LMP is 4.9% better than RoG on WebQSP and is 13.9% better than KG-Agent on CWQ. It is worth mentioning that LMP does not need parameter learning and is completely zero-shot (i.e., don't need any example designed by human). Compared with other prompting counterparts, LMP also leads by a significant margin. Even with Llama-2-70B or Llama-3-70B, LMP still has comparable or better performance than other methods with GPT-4.

For Q2, we examine how different backbone LLMs affect LMP in Table 2 and add Llama-3-8B and o1-mini as backbone LLMs to further investigate the impact of smaller and more capable recent LLMs. We discover that LMP also performs relatively well with smaller LLMs, where it outperforming any baselines with more capable larger LLMs in WebQSP dataset and is only behind KG-enhanced methods with GPT-4 in CWQ dataset. LMP harnesses the inherent strengths of LLMs in text summarization and comprehension, tasks that even smaller LLMs are well capable of, rather than more demanding tasks such as numerical scoring of triples for path exploration in ToG, which less capa-

with ChatGPT	LMP					CoT
Noise Level	0%	20%	40%	60%	80%	-
WebQSP	87.2	84.5	82.8	80.3	79.2	62.2
CWQ	75.2	69.0	64.1	61.5	57.1	38.8

Table 3: The exact match accuracy (%) of LMP under different levels of noise in KG.

Models	WebQSP		CWQ	
	# calls	# tokens	# calls	# tokens
ToG	16.7	6351	25.6	7935
FiDeLiS	10.7	2452	15.2	2741
LMP	**5.3**	**1564**	**9.5**	**2474**

Table 4: The average number of calls and token usage for LLM (ChatGPT) per question.

	CWQ	GrailQA
LMP without aggregation	72.6	78.8
LMP without transformation	73.8	80.4
LMP without readout	67.0	64.0
LMP-path	64.0	75.0
LMP	**75.2**	**82.6**

Table 5: The EM accuracy (%) of the variants of LMP.

ble smaller LLMs often struggle with. Furthermore, LMP's performance rises alongside more capable LLMs with increasing question answering abilities, where LMP still achieves a 21.8% and 53.5% improvement with more recent o1-mini compared to CoT with it. Interestingly, we discover that LMP with the least capable Llama-2-70B among backbone LLMs still beat CoT with the most capable o1-mini. Overall, LMP gains average 51.5% and 20.5% performance improvements over baselines without and with external knowledge respectively.

4.3 Robustness Analysis (Q3)

To simulate the real-world scenario where KGs are often incomplete and noisy, we randomly remove a certain percentage of relations and entities names in KG. For unnamed entities, we perform additional aggregation operation to gather their neighborhood information as surrogates for their names, This process does not rely on LLMs and thus does not introduce additional complexity to our method. As shown in Table 3, the increasing noise and incompleteness of KG have relatively small impact on LMP, showing its great robustness empowered by aggregation. Moreover, even with 80% of noise, LMP still gains an average 37.5% improvement over vanilla LLM. The reason behind this is quite intriguing that our aggregation and transformation combination only need a small amount of related information to make an accurate prediction on the missing pieces since it gain an overall understanding of the graph. Take the example of missing 48 entity names out of 50 about states in USA, in raw triples format *(unknown, contains, California), (unknown, contains, unknown), (unknown, contains, Florida), ...*, it is unclear what "unknown" represents and whether the "unknown" head entities refer to the same thing. But when we aggregate triples in neighborhoods fashion *(unknown, contains, [California, unknown, Florida, ...])*, it becomes evident that the "unknown" head entity is the USA and any "unknown" in tail entities are the states of USA. Transformation leverages the inherent knowledge of LLM and fills out these missing pieces with ease, which offers our method great robustness in real-world scenario where KGs are often incomplete.

4.4 Efficiency Analysis (Q4)

LMP inherently has efficiency advantages over fine-tuning methods since it is a train-free plug-and-play framework, and also excels in prompting methods, where LMP has the least call number and token usage as shown in Table 4. Let search width and depth as K and L respectively. LMP calls LLM $(2L+1)$ times per question for sampling and transformation in each layer and plus the final question answering, which is much more efficient than ToG $(2LK+K+1)$ and FiDeLis $(LK+K+1)$. Notably, the complexity of these path-based methods is width K dependent since they need to explore each path separately, whereas LMP whose complexity is width K irrelevant since LLM can summarize multiple facts at once. As for tokens usage, LMP is also tokens efficient thanks to aggregation operation that saves up to $2/3$ tokens than path-based methods using raw paths which contain the redundant head entities and relations names. Moreover, methods like ToG and FiDeLis use in-context exemplars to guide the path exploration, whereas LMP is zero-shot leveraging the natural strength of LLM in text summarization.

4.5 Ablation and Hyperparameters (Q5)

We conduct ablation studies with ChatGPT as backbone LLM in Table 5 and we (1) remove aggregation and send raw triples into LLM; (2) remove transformation and augment LLM with raw aggregated neighborhood information; (3) remove

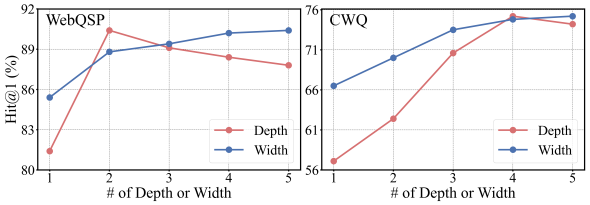

Figure 3: Hyperparameters analysis of LMP.

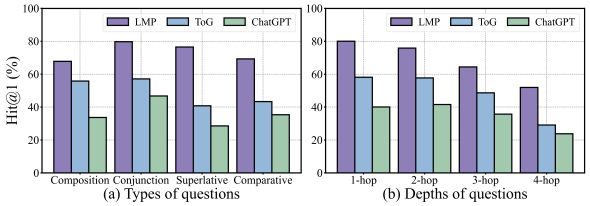

Figure 4: The performance of methods on various types (a) and depths (b) of questions on CWQ dataset.

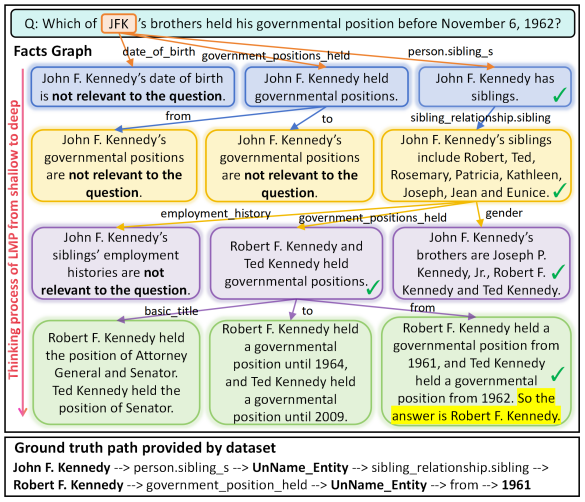

Figure 5: The thinking process of LMP with facts graph.

readout and use the last layer of language message passing to augment LLM. Each component shows a unique contribution to the overall performance. In addition, to validate our semantic summarized facts vs. keyword-based paths inputs, we augment LLM with the triples mentioned in facts graph as LMP-path to ensure both approaches have the same extracted knowledge from KG. The performance of keyword-based LMP-path downgrades 11% on CWQ and 8% on GrailQA, which indicates that the raw keyword-based paths are unorganized and semantically awkward for LLMs. We conduct additional experiments on the error propagation and the impact of the previous stage on its subsequent stage in Appendix D.3.

We also analyze the hyperparameters influence of LMP in Figure 3, where the performance increases at first with depth because more useful information is explored. Increasing the depth too much leads to a decline in performance, as the information that is too distant from the topic entity becomes irrelevant and introduces noise. As for width, the performance improvements become more and more slim with width increasing.

4.6 Types and depths of the question (Q6)

We present the performance of vanilla LLM, ToG, and LMP with ChatGPT on composition, conjunction, superlative, and comparative types of questions in CWQ dataset. In Figure 4a, LMP has huge performance improvement over vanilla LLM and ToG on all types of questions, especially on superlative and comparative questions that require information from all possible candidates to answer. This demonstrates that the facts graph of LMP is more informative in representing the knowledge from KG than the paths used by ToG.

We further investigate how LMP performs on different depths of questions in Figure 4b, where the complexity and difficulty of the question increase along with the depth. On the shallow depth questions of 1-hop and 2-hop that emphasize the ability of knowledge extraction, LMP outperforms vanilla LLM by about 40% and path-based ToG by 20%, indicating LMP is able to accurately locate and extract needed knowledge from KG. On the deep questions of 3-hop and 4-hop that emphasize the step-wise reasoning ability, LMP still outperforms vanilla LLM by about 30% and path-based ToG by 20%, which proves the workflow of LMP that digesting information from shallow to deep progressively to be crucial in solving real-world complex problems.

4.7 Human-level Explainability (Q7)

In Figure 5, we showcase the explainability of our facts graph with a question from CWQ dataset, which asks about *Which of JFK's brothers held his governmental position before November 6, 1962*. For humans the above complex question can be naturally decomposed into sub-questions that are easier to solve: finding JFK's brothers, who holds governmental position, and the duration of that position. The facts graph also exhibits such ability in Figure 5 from top to bottom, showcasing the process of LLM breaking down a complex question and accordingly searching and digesting information on KG from shallow to deep progressively. At depth 1 and 2 on KG (blue and yellow boxes), LMP

is trying to find the JFK's siblings and locate 8 of them. Notably, LMP identifies the irrelevant information (boldface text) while summarizing facts and stops further exploration. And then at depth 3 (purple boxes), LMP finds 3 of 8 siblings are male who are brothers of JFK and 2 of them held governmental positions. Lastly at depth 4 (green boxes), LMP learns the start time of those positions and concludes the correct answer for the question (highlighted text). Compared with our facts graph that contains the complete thinking process of LLM solving the question, the path in the bottom part of Figure 5, even though it is correct, only provides us the keywords of reaching the correct answer on KG and thus offers limited explainability. We provide additional case studies in Appendix E.3.

5 Conclusion

In this paper, we propose Language Message Passing that iteratively aggregates and transforms neighborhood information into summarized facts and then augments LLM with topological readout from the facts graph where nodes are summarized facts. LMP achieves the SOTA performance on all 5 knowledge graphs question answering datasets and offers human-level semantic explainability.

6 Limitations

We consider that our limitations are mainly in two aspects. Firstly, due to budget constraints, we only used the Llama from Meta and GPT series from OpenAI in our experiments for open source and close API LLMs. However, proposed LMP is a universal framework which can be equipped with various LLMs. Secondly, the topic of this paper focuses on KG-enhanced RAG, and our setting concentrates on the scenarios where a KG is provided. Although we demonstrates the practical applicability of LMP in real-world scenarios, there are still various types of knowledge sources such as documents and websites that need to also be considered in real-world scenarios. We leave exploration with other LLMs and incorporating LMP with other types of knowledge source as future work.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Jinheon Baek, Alham Fikri Aji, and Amir Saffari. 2023. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. *arXiv preprint arXiv:2306.04136*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*, pages 1533–1544.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *NeurIPS*, pages 1877–1901.

Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. 2018. Harp: Hierarchical representation learning for networks. In *AAAI*, pages 2127–2134.

Zixuan Dong, Baoyun Peng, Yufei Wang, Jia Fu, Xiaodong Wang, Xin Zhou, Yongxue Shan, Kangchen Zhu, and Weiguo Chen. 2025. Effiqa: Efficient question-answering with strategic multi-model collaboration on knowledge graphs. In *COLING*, pages 7180–7194.

Haishuo Fang, Xiaodan Zhu, and Iryna Gurevych. 2024. DARA: Decomposition-alignment-reasoning autonomous language agent for question answering over knowledge graphs. In *ACL*, pages 3406–3432.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *ICML*, pages 1263–1272.

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond iid: three levels of generalization for question answering on knowledge bases. In *WWW*, pages 3477–3488.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*, pages 1024–1034.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*.

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 preprint arXiv:2402.11163*.

Xinke Jiang, Ruizhe Zhang, Yongxin Xu, Rihong Qiu, Yue Fang, Zhiyuan Wang, Jinyi Tang, Hongxin Ding, Xu Chu, Junfeng Zhao, et al. 2023. Think and retrieval: A hypothesis knowledge graph enhanced medical large language models. *arXiv preprint arXiv:2312.15883*.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2024. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *ICLR*.

Yuan Sui, Yufei He, Nian Liu, Xiaoxin He, Kun Wang, and Bryan Hooi. 2024. Fidelis: Faithful reasoning in large language model for knowledge graph question answering. *arXiv preprint arXiv:2405.13873*.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-graph: Deep and responsible reasoning of large language model with knowledge graph. In *ICLR*.

Yueqing Sun, Qi Shi, Le Qi, and Yu Zhang. 2022. Jointlk: Joint reasoning with language models and knowledge graphs for commonsense question answering. In *NAACL*, pages 5049–5060.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *ACL*, pages 641–651.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.

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

Yuqi Wang, Boran Jiang, Yi Luo, Dawei He, Peng Cheng, and Liangcai Gao. 2024. Reasoning on efficient knowledge paths: Knowledge graph guides large language model for domain question answering. *arXiv preprint arXiv:2404.10384*.

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. In *NeurIPS*, pages 24824–24837.

Yilin Wen, Zifeng Wang, and Jimeng Sun. 2024. Mindmap: Knowledge graph prompting sparks graph of thoughts in large language models. In *ACL*.

Yike Wu, Nan Hu, Guilin Qi, Sheng Bi, Jie Ren, Anhuan Xie, and Wei Song. 2023. Retrieve-rewrite-answer: A kg-to-text enhanced llms framework for knowledge graph question answering. *arXiv preprint arXiv:2309.11206*.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks? In *ICLR*.

Yao Xu, Shizhu He, Jiabei Chen, Zihao Wang, Yangqiu Song, Hanghang Tong, Kang Liu, and Jun Zhao. 2024. Generate-on-graph: Treat llm as both agent and kg in incomplete knowledge graph question answering. *arXiv preprint arXiv:2404.14741*.

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *ACL*, pages 201–206.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. In *NeurIPS*, pages 4805–4815.

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Wang, Zhiguo Wang, and Bing Xiang. 2023. Decaf: Joint decoding of answers and logical forms for question answering over knowledge bases. In *ICLR*.

Donghan Yu, Chenguang Zhu, Yiming Yang, and Michael Zeng. 2022. Jaket: Joint pre-training of knowledge graph and language understanding. In *AAAI*, pages 11630–11638.

Qinggang Zhang, Junnan Dong, Hao Chen, Xiao Huang, Daochen Zha, and Zailiang Yu. 2023a. Knowgpt: Black-box knowledge injection for large language models. *arXiv preprint arXiv:2312.06185*.

Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D Manning, and Jure Leskovec. 2022. Greaselm: Graph reasoning enhanced language models for question answering. In *ICLR*.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. 2023b. Siren's song in the ai ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*.

Ruilin Zhao, Feng Zhao, Long Wang, Xianzhi Wang, and Guandong Xu. 2024. Kg-cot: Chain-of-thought prompting of large language models over knowledge graphs for knowledge-aware question answering. In *IJCAI*, pages 6642–6650.

Appendices

A Implementation Details of LMP

For reproducibility, here we present the implementation details of LMP, including the algorithms, prompts, and constraint enforcement mechanism with retry we used.

A.1 Algorithm for Language Message Passing

We show the detailed algorithm of Language Message Passing in Algorithm 1. Note since LLM can summarize multiple facts at the same time, the aggregation and transformation for each layer as in lines 8 and 9 are processed only once and we write in a such way for better clarification purpose.

Algorithm 1: Language Message Passing

Input: question q, KG $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{X})$, topic entity E_q
Parameter: width K, depth L
Output: answer of question q

1 Initialize $\hat{\mathcal{E}}$, $\hat{\mathcal{R}}$, $\mathcal{H}$;
2 **for** $l \in L$ **do**
3 Get $(E^{(l)}, X^{(l)}, R^{(l)})$ of E_q in $\mathcal{G}$;
4 Initialize $\hat{E}^{(l)}$, $\hat{R}^{(l)}$, $H^{(l)}$;
5 $\hat{R}^{(l)} \leftarrow R^{(l)}$ using Eq. (1);
6 $\hat{\mathcal{R}} \leftarrow \hat{\mathcal{R}} \cup \hat{R}^{(l)}$;
7 **for** $k \in K$ **do**
8 $a_k^{(l)} \leftarrow (X^{(l-1)}, \hat{r}_k^{(l)}, X_k^{(l)})$ using Eq. (2) ;
9 $h_k^{(l)} \leftarrow a_k^{(l)}$ using Eq. (3) ;
10 $\hat{e}_k^{(l)} \leftarrow E_k^{(l)}$ using Eq. (4) ;
11 $\hat{E}^{(l)} \leftarrow \hat{E}^{(l)} \cup \hat{e}_k^{(l)}$;
12 $H^{(l)} \leftarrow H^{(l)} \cup h_k^{(l)}$;
13 **end**
14 $\hat{\mathcal{E}} \leftarrow \hat{\mathcal{E}} \cup \hat{E}^{(l)}$;
15 $\hat{\mathcal{H}} \leftarrow \hat{\mathcal{H}} \cup H^{(l)}$;
16 **end**
17 $z \leftarrow \hat{\mathcal{G}} = (\hat{\mathcal{E}}, \hat{\mathcal{R}}, \mathcal{H})$ using Eq. (5) ;
18 **return** answer $= \mathrm{LLM}(Z, q)$

A.2 Algorithm for Topological Readout

We show the detailed algorithm of Topological Readout in Algorithm 2. Note the "+" signs in line 10 and 18 mean text concatenation.

Algorithm 2: Topological Readout

Input: facts graph $\hat{\mathcal{G}} = (\hat{\mathcal{E}}, \hat{\mathcal{R}}, \mathcal{H})$, topic entity e_q
Output: Summary Outlines Z

1 Initialize stack S, index dict D;
2 S.put(e_q) ;
3 $D[e_q] =$ ' ' ;
4 **while** S is not null **do**
5 $top = S$.pop() ;
6 **if** top.neighbors() is not null **then**
7 $idx = 1$;
8 **for** e_n in top.neighbors() **do**
9 **if** e_n is not visited **then**
10 $D[e_n] = D[top] + idx$.str() +'.' ;
11 S.put(e_n) ;
12 $idx += 1$;
13 **end**
14 **end**
15 **end**
16 **end**
17 **for** e_i in D.keys() **do**
18 $z_i = D[e_i] + h_i$
19 **end**
20 **return** Z

A.3 Sampling Prompt

The sampling prompt for the first layer of language message passing is slightly different from deeper layers, since there is no summarized fact from previous layer at the first layer. The sampling prompts are listed as follows:

Sampling prompt at layer 1 of LMP

Given the question, we have the topic of the question and its relations.
question: ⟨Question⟩
topic: ⟨Topic⟩

Please select top-K relations from the options below to explore about the topic to answer the question and just return selected relations in a numbered list without explanation.
options: ⟨Relations⟩

where ⟨Question⟩ is the question to answer and ⟨Topic⟩ is the topic entity name. The ⟨Relations⟩ is the relations we want to reduce and only keep

top-K most relevant ones.

Sampling prompt at layer 2+ of LMP

Given the question and its topic, we have some summarized facts and their relations.
question: ⟨Question⟩
topic: ⟨Topic⟩

Please select top-K relations from the options below to explore about the topic to answer the question and just return selected relations in a numbered list without explanation.
1. fact: ⟨Fact_1⟩ options: ⟨Relations_1⟩
2. fact: ⟨Fact_2⟩ options: ⟨Relations_2⟩
...
K. fact: ⟨Fact_K⟩ options: ⟨Relations_K⟩

Since we select top-K relations from previous layer, we also have K summarized facts from each one of them. Here we present ⟨Fact⟩ as each summarized fact with its corresponding relations ⟨Relations⟩.

A.4 Aggregation Prompt

For each head entity and its connected relation, we aggregate the neighbor entities all at once as follows:

Aggregation prompt of LMP

The entity ⟨HeadEntity⟩ **has relation** ⟨Relation⟩ **with following entities:** ⟨TailEntity_1⟩, ⟨TailEntity_2⟩, ..., ⟨TailEntity_n⟩.

where ⟨HeadEntity⟩ is the head entity with its connected relation ⟨Relation⟩, and one head entity can be connected to multiple tail entities ⟨TailEntity_1⟩, ⟨TailEntity_2⟩, ..., ⟨TailEntity_n⟩ through one relation.

A.5 Transformation Prompt

Similar to the sampling prompt, our transformation prompt for the first layer of language message passing is also slightly different from deeper layers, since there is no summarized fact from previous layer at the first layer. The transformation prompts are listed as follows:

Transformation prompt at layer 1 of LMP

Given the question, we have K facts about its topic and related relations that may helpful to answer the question.
question: ⟨Question⟩
topic: ⟨Topic⟩

Please summarize each following fact while only keeping every relevant information about the question and just return all summarized facts as following order in the same numbered list without explanation.
facts:
1. ⟨$\text{AgregatedInformation}_1$⟩
2. ⟨$\text{AgregatedInformation}_2$⟩
...
K. ⟨$\text{AgregatedInformation}_K$⟩

where ⟨Question⟩ is the question to answer for our task and ⟨Topic⟩ is its topic entity name. We transform the aggregated neighborhood information AgregatedInformation for each relation in top-K selected relations.

Transformation prompt at layer 2+ of LMP

Given the question, we have K facts with some background information related to them and the topic.
question: ⟨Question⟩
topic: ⟨Topic⟩
background: ⟨Facts⟩

Please summarize each following fact while only keeping every relevant information about the question and just return all summarized facts as following order in the same numbered list without explanation.
facts:
1. ⟨$\text{AgregatedInformation}_1$⟩
2. ⟨$\text{AgregatedInformation}_2$⟩
...
K. ⟨$\text{AgregatedInformation}_K$⟩

where ⟨Facts⟩ are the summarized facts from previous layer of language message passing. Note only the facts that associated with the selected top-K relations will be mentioned here.

A.6 Question Answering Prompt

Question Answering of LMP

Based on the given the facts and your own knowledge, please the answer the question as simple as possible and only return all the possible answers in a numbered list.
facts: ⟨SummaryOutlines⟩
question: ⟨Question⟩

where ⟨SummaryOutlines⟩ are the summary outlines that readout from the all the facts graph for each topic entity after multi-layer of language message passing.

A.7 Constraint Enforcement Mechanism with Retry

In case of LLM fails to accomplish the operations in LMP, we implement a retry strategy for sampling and transformation operations, where the maximum retry limits are set to 5 and temperature of LLMs increases 0.2 per retry. If the output of LLM still fails to meet our requirement after three times of retries, we directly ask LLM to answer the question without knowledge augmentation.

We also implement constraint enforcement mechanism to ensure that LLM works as we intended. For sampling, we ask LLM to select top K question-related $\hat{R}^{(l)}$ relations and return in a numbered list. If LLM returns less than K relations, we combine the selected relations in this attempt with previous attempts until K relations are met or maximum retry limits are reached. For transformation, we ask LLM to summarize given K aggregated neighbor information and return K summarized facts in a numbered list. If the number of summarized facts returned is not equal to K, we proceed with a retry. Note for the final question answering, we directly match the ground truth answers from the output of LLM without any constraint enforcement.

Since we leverage the strengths of LLM in context understanding and summarization, our method is completely zero-shot with minimal instructed constraint and enforcement mechanism. In practice, the retry mechanism is rarely triggered. We show the overall retry numbers of LMP with weaker LLM and stronger LLM (Llama-3-8B & 70B) in Table 6 and discover no significant difference, which validates the effectiveness of our methods that even small and weaker LLMs are well capable of.

# of retry per question	WebQSP	CWQ
LMP + Llama-3-8B	0.214	0.453
LMP + Llama-3-70B	0.199	0.419

Table 6: The number of retry triggered per question.

B Details of Baselines

We compare our method with three types of baselines, namely LLMs only, fine-tuning and prompting.

B.1 LLMs Only Methods

LLMs only methods use LLMs' own knowledge to answer questions, without using any external knowledge. Specifically, we compare with the following methods:

- **Standard prompting (IO prompt)**. It gives LLMs some specific input-output pairs as examples to help them understand instruction.
- **Chain of thought prompting (CoT)**. CoT uses prompt to make LLMs generate intermediate steps or explanations of questions to arrive at more accurate answers.
- **Self-consistency (SC)**. It leverages the intuition that a complex reasoning problem typically admits multiple different ways of thinking leading to its unique correct answer. Specifically, It first samples a diverse set of reasoning paths, and then selects the most consistent answer by marginalizing out the sampled reasoning paths.

B.2 Fine-tuning Methods

Fine-tuning methods modify parameters of LMs or LLMs, which are less efficient. Specifically, we compare with the following methods:

- **DECAF**. It jointly generates both logical forms and direct answers, and then combines the merits of them to get the final answers.
- **RoG**. It synergizes LLMs with KGs to conduct faithful and interpretable reasoning. Specifically, it presents a planning-retrieval-reasoning framework, which allows LLMs to access the latest knowledge while reasoning based on faithful plans on graphs.
- **KG-Agent**. It proposes an autonomous agent framework to synergize LLMs and KGs to perform complex reasoning over KG.

- **Retrieve-Rewrite-Answer**. It proposes an answer-sensitive KG-to-Text approach that can transform KG knowledge into well-textualized statements, which are more suitable for KGQA.

B.3 Prompting Methods

Prompting methods don't need to conduct any parameters learning for LLMs, which is much more efficient. LMP proposed by us also belongs to this type, and we compare it with the following methods:

- **ToG**. It uses LLMs as agent to iteratively execute beam search on KG. Specifically, it selects relation and entity alternately to discover reasoning paths, which are then used as external knowledge for LLMs answering.

- **ToG-R**. It is a variant of ToG, which only uses LLMs to select relations. For entities, it uses random prune to mitigate the risk of misguided reasoning and reduce the overall cost.

- **EffiQA**. It proposes a new integration paradigm of LLMs and KGs for multi-step reasoning. Through an iterative paradigm of global LLM planning, efficient KG exploration, and self-reflection, it balances leveraging LLM capabilities with maintaining computational efficiency.

- **FiDeLis**. It proposes a retrieval-exploration interactive method to enhance intermediate steps of LLM reasoning grounded by KGs. The Path-RAG module and the use of deductive reasoning as a calibration tool effectively guide the reasoning process, leading to more accurate knowledge retrieval and prevention of misleading reasoning chains.

- **KG-CoT**. It leverages a small-scale step-by-step graph reasoning model to reason over KGs and utilizes a reasoning path generation method to generate chains of knowledge with high confidence for large-scale LLMs.

C Details of Experiment Settings

C.1 Detailed Version of Backbone LLMs

We use Llama-2-70B (Llama-2-70b-chat), Llama-3-70B (Meta-Llama-3-70B-Instruct), ChatGPT (gpt-3.5-turbo-0125), GPT-4 (gpt-4-0613), Llama-3-8B (Meta-Llama-3-8B-Instruct), and o1-mini (o1-mini-2024-09-12) as backbone LLMs.

C.2 Detailed Hyperparameters Settings

In Table 7, we listed the hyperparameters settings of width K and depth L for each dataset to reproduce our experiment results in Section 4.2 (Q2). To avoid confusion and align with baselines, the depth L here means the depth of information searched from the topic entities. Since LMP implements an additional aggregation operations for unnamed entities, our method only need to implement $(L-1)$ round of language message passing to explore L-hop information of the topic entity.

C.3 Detailed SPARQL for Unnamed Entity

In real-world scenarios, KGs are often incomplete and noisy. There are many entities have no text descriptions in Freebase. To address this, we perform addition aggregation and use the text descriptions of its 1-hop neighbor entities as its text description for unnamed entities instead of using *unknown* tag or special tokens like *UnName_Entity* in ToG to indicate the missing entity name. This trick does not increase the calls for LLMs. The SPARQL query for entity name search in our method is detailed as below:

```
PREFIX ns: <http://rdf.freebase.com/ns/>
SELECT DISTINCT ?name ?extra
WHERE {
VALUES ?start {%s}
?start ns:%s ?e .
FILTER(?e != ns:%s)
OPTIONAL {?e ns:type.object.name ?name .}.
OPTIONAL
{FILTER (!BOUND(?name))
{?e ?r ?extra . FILTER (isLiteral(?extra))}
UNION
{?e ?r ?e1 . ?e1 ns:type.object.name ?extra .}} .
}
```

where *?name* searches for the original entity name, *?extra* searches for the neighboring entity names of the unnamed entity.

D Additional Experiments

D.1 Additional F1 Score Performance

We show the additional F1 performance in Table 8. Note for the F1 evaluation, we change the phrasing in the final question answering prompt and remove the part of "return all the possible answers" to avoid LLM generating multiple aliases for one answer, which hurts the precision score. We put the F1 evaluation in the Appendix since there are only few related works reporting F1 scores and the reasons behind this are as followed. First, RAG methods use LLM to output a paragraph of texts and without a specific instructed prompt it is difficult to determined how many answers are in the

Backbone LLMs	WebQSP		CWQ		GrailQA		Simple Questions		WebQuestions	
	Width	Depth	Width	Depth	Width	Depth	Width	Depth	Width	Depth
Llama-2-70B	5	2	3	4	3	3	5	1	5	2
Llama-3-70B	5	2	3	4	3	3	5	1	5	2
ChatGPT	5	2	5	4	5	3	5	1	5	2
GPT-4	5	2	5	4	5	3	5	1	5	2

Table 7: The hyperparameters settings of width K and depth L of our experiments results in Section 4.2 (Q1) for each dataset.

Model		WebQSP	CWQ
LLMs only	CoT + ChatGPT	54.7	35.8
Fine-tuning	DECAF	78.8	-
	RoG	70.8	56.2
Prompting	ToG + ChatGPT	72.3	56.9
	FiDeLiS + ChatGPT	76.7	61.7
	LMP + ChatGPT	**79.3**	**64.5**

Table 8: The F1 score (%) of different models.

Models		WebQSP	CWQ
ToG	with BM25	58.7	51.4
LMP		**75.1**	**60.4**
ToG	with SentenceBERT	66.3	51.7
LMP		**80.3**	**62.1**
ToG	with ChatGPT	76.2	58.8
LMP		**87.2**	**72.6**

Table 9: The exact match accuracy (%) with different relation sampling methods.

response. Second, unlike the traditional semantic parsing methods that execute generated SPARQL queries as the final QA stage and the space of possible answers is within the ground truth answers, RAG methods use LLM to output answers that are far beyond the range of ground truth answers. The comparison between them may seem unfair (Fang et al., 2024). And last, the ground truth answers in datasets could be overly restrictive, making it difficult to determine the true precision of an answer. These datasets define ground truth answers at a fixed depth of the topic entity in the KG. For example, the question is "where was XXX born?" and the answer in KG is "Los Angeles". However, "USA" and "California" are connected to the ground truth answer entity "Los Angeles" with relation "location.location.containedby" in KG and are also valid answers for the question.

D.2 Different sampling methods

Our sampling operation uses LLM to select most relevant relations of based on the topic and question. This operation can also be implemented using lightweight models that measure text similarity such as BM25 or SentenceBERT. As mentioned in (Sun et al., 2024), the answer does not necessarily show text similarity with the question, so we use LLM to perform relation sampling to achieve the best performance. We show the performance with different relation sampling methods in Table 9, where LLM has clear edge over other methods.

D.3 Error Propagation

The overall framework of LMP consists multiple stages, where sampling, transformation, and final QA stages involve using LLM. We conduct a detailed experiment to examine if errors occur in each LLM-involved stage and their impact on subsequent stages with Llama-3-8B (smaller & weaker) and Llama-3-70B (larger & stronger). Given the recursive nature of our method, we investigate each stage in the last-hop of its exploration.

We first examine errors in sampling stage by comparing relations sampled with the ground truth SPARQL queries in Table 10. Round 50% of the questions in simple WebQSP (1- & 2-hop) and 20% of the questions in complex CWQ (3- & 4-hop) dataset retrieve all the correct relations. If all correct relations are retrieved, the answer entity is already reached. The overall exact match accuracy of our method (about 89% and 70% for WebQSP and CWQ) is consistently higher than the percentage of all correct relations sampled. This demonstrates the importance of following stages and their abilities to correct the errors in relation sampling. But overall, relation sampling retrieves certain answer-related information from KG in about 90% of the questions across both datasets.

We further investigate the following transformation by examining whether summarized facts contain the answers in Table 11, given all/partial/no correct relation sampled. We discover that given all correct relations the transformation rarely in-

% of question	all correct relations sampled		partial correct relations sampled		no correct relations sampled	
	WebQSP	CWQ	WebQSP	CWQ	WebQSP	CWQ
LMP + Llama-3-8B	50.6	19.6	36.5	66.6	12.8	13.8
LMP + Llama-3-70B	53.9	26.9	40.9	65.5	5.1	7.6

Table 10: The percentage of question with all/partial/no correct relations sampled.

EM (%) of facts	all correct relations sampled		partial correct relations sampled		no correct relations sampled	
	WebQSP	CWQ	WebQSP	CWQ	WebQSP	CWQ
LMP + Llama-3-8B	96.5	95.3	86.6	77.7	34.7	28.8
LMP + Llama-3-70B	97.3	95.5	88.5	83.8	44.0	33.7

Table 11: The exact match accuracy (%) of facts with all/partial/no correct relations sampled.

EM (%) of final QA	w. correct facts		w/o. correct facts	
	WebQSP	CWQ	WebQSP	CWQ
LMP + Llama-3-8B	95.5	91.7	56.6	22.4
LMP + Llama-3-70B	96.1	94.3	58.0	24.5

Table 12: The exact match accuracy (%) of final QA with and without correct facts.

troduces error (96%+ EM), with minor deviations due to phrasing variations of the LLM's inherent preference (e.g., "US president" vs. "president of united states"). With partial correct relations, transformation still produces answer-containing facts in about 80% of cases thanks to the knowledge in KG and in LLM complement each other. Take the example question "which of JFK's brother held government position". Given the sibling of JFK from KG, LLM can identify their gender based on their names (name robert tends to be male) despite of relation gender is not retrieved after sampling. The example in our case study Section 4.7 shows transformation can identify irrelevant facts by stopping summarizing with "not relevant" tag on it, which stops the error in previous stage propagates into next stage. Lastly, with no correct relation 30% of facts still contain the answer since LLM leverages its own knowledge to reach the answer.

We investigate how final QA performs with the correct facts (answer inside) and without correct facts in Table 12. Given the correct facts, about 95% in WebQSP and 93% in CWQ of question are correctly answered by final QA. In most cases, it is due to the question can be answered in multiple ways but the ground truth answers only validate for one of them. And with no correct facts, LLM leverages its own inherent knowledge and correctly answers 57% and 23% of questions in WebQSP and CWQ without the external KG knowledge.

To summarize all: 1).With the correct previous stages, the following stages are also mostly correct (about 95%), which ensures the entire pipeline of our work operates seamlessly. 2).The subsequent stages can correct the error in previous stages and the failure in one stage does not necessarily affect the final results. 80% of the transformation are correct (output facts with answer inside) without all correct relation sampling from previous stage. 3).The weaker LLM (Llama-3-8B) is well capable of the operations in our proposed methods. The performance drop-off from Llama-3-70B to 8B is acceptable given the difference in inherent abilities of the LLMs.

D.4 Additional Case Studies

We show additional case studies in Table 13, 14, 15 to emphasize the strength of LMP in terms of explainability. We can clearly see that our summary outlines are much more informative than the reasoning paths, which contain the whole thinking process of LMP. As in Table 13, the reasoning path only shows us the correct answer while our facts graph is able to discover *Arthur Miller* is influenced by three people and only *William Shakespeare* influenced by *Lucian*, which matches the question. As in Table 15, the reasoning path is unable to find the answer while our facts graph showcases the complete logical chain of solving the problem, which identifies the college of *Sampson Salter Blowers* and its state. Additionally, KG like Freebase has some unnamed entity as shown in reasoning paths, which diminishes their explainability. As for our summary outlines in LMP, those unnamed entities can be inferred from context information by LLM itself, which provides a better explainability.

Question	Who influenced Arthur Miller that was influenced by Lucian?
Reasoning Paths in ToG	(Path 1, Score: 0.75) Arthur Miller → *influence.influence_node.influenced_by* → William Shakespeare → *influence.influence_node.influenced_by* → Lucian (Path 2, Score: 0.2) Lucian → *influence.influence_node.influenced_by* → Socrates → *influence.influence_node.influenced_by* → Parmenides (Path 3, Score: 0.05) Arthur Miller → *people.person.education* → UnName_Entity → *education.education.student* → Arthur Miller
Summarized Facts in LMP	1. Arthur Miller was influenced by Henrik Ibsen, Sophocles, and William Shakespeare. 1.1. William Shakespeare was influenced by Lucian, who influenced someone who influenced Arthur Miller. 1.2. Henrik Ibsen influenced many people, but none of them influenced Arthur Miller. 1.2.1. William Shakespeare was influenced by Lucian, who influenced someone who influenced Arthur Miller. 1.2.2. August Strindberg was influenced by William Shakespeare. 1.2.3. Henrik Ibsen did not influence anyone who influenced Arthur Miller. 1.3. The education of William Shakespeare is not relevant to the question. 2. Arthur Miller's professions include Actor, Essayist, Playwright, Screenwriter, and Voice Actor. 3. Arthur Miller attended Abraham Lincoln High School and University of Michigan, earning a High School Diploma and Bachelor's degree, respectively.
Ground Truth	William Shakespeare

Table 13: Additional Case Study for explainability in Section 4.6 (Q5)

Question	What is the state where the team whose fight song is "Renegade" is from?
Reasoning Paths in ToG	(Path 1, Score: 0.67) Renegade → sports.fight_song.sports_team → Pittsburgh Steelers (Path 2, Score: 0.33) Renegade → sports.sports_team.fight_song → UnName_Entity
Summarized Facts in LMP	1. The Renegade is the fight song of the Pittsburgh Steelers. 1.1. The Pittsburgh Steelers is located in Pittsburgh. 1.1.1. Pittsburgh is located in Pennsylvania. 1.1.2. Pittsburgh is home to the Pittsburgh Steelers. 1.2. The division of Pittsburgh Steelers is not relevant to the question. 1.3. The fight songs of Pittsburgh Steelers also include "Black and Yellow", "Here We Go", and "Steelers Polka". 1.3.1. The composers of these song are not relevant to the question. 2. The Renegade has multiple recordings. 3. The Renegade is an album.
Ground Truth	Pennsylvania

Table 14: Additional Case Study for explainability in Section 4.6 (Q5)

Question	What state is the college that Sampson Salter Blowers is a grad student of located?
Reasoning Paths in ToG	(Path 1, Score: 0.75) Sampson Salter Blowers → *education.education.student* → UnName_Entity → *education.education.institution* → Harvard College (Path 2, Score: 0.2) Sampson Salter Blowers → *education.education.student* → UnName_Entity → *education.educational_institution.students_graduates* → { } (Path 3, Score: 0.05) Sampson Salter Blowers → *education.education.student* → UnName_Entity → *people.person.education* → { }
Summarized Facts in LMP	1. Sampson Salter Blowers attended Harvard College. 1.1. Harvard College is the institution Sampson Salter Blowers attended. 1.1.1. Harvard College is located in Cambridge, Massachusetts, United States of America. 1.1.2. Harvard College has a campus in Harvard College. 1.1.3. Harvard College is a part of Harvard University. 2. Sampson Salter Blowers is Canadian. 2.1. The locations Canada contained are not relevant to the question. 3. Sampson Salter Blowers was born in Boston. 3.1. The locations Boston contained are not relevant to the question.
Ground Truth	Massachusetts

Table 15: Additional Case Study for explainability in Section 4.6 (Q5)