Struct-X: Enhancing the Reasoning Capabilities of Large Language Models in Structured Data Scenarios

Xiaoyu Tan*
txywilliam1993@outlook.com
INF Technology (Shanghai) Co., Ltd.
Shanghai, China

Haoyu Wang*
wanghy@sues.edu.cn
Shanghai University of Engineering Science
Shanghai, China

Xihe Qiu*†
qiuxihe1993@gmail.com
Shanghai University of Engineering Science
Shanghai, China

Leijun Cheng
leijuncheng@sues.edu.cn
Shanghai University of Engineering Science
Shanghai, China

Yuan Cheng
cheng_yuan@fudan.edu.cn
AI^3 Institute, Fudan University
Shanghai, China

Wei Chu
chuwei@inftech.ai
INF Technology (Shanghai) Co., Ltd.
Shanghai, China

Yinghui Xu
xuyinghui@fudan.edu.cn
AI^3 Institute, Fudan University
Shanghai, China

Yuan Qi
qi.yuan@outlook.com
AI^3 Institute, Fudan University
Shanghai, China

Abstract

Conducting reasoning tasks with large language models (LLMs) on structured and redundant data poses significant challenges, primarily due to the complexity introduced by the structured markdown tokens and the presence of extraneous contextual information. These elements can overburden and disrupt the generation process of LLMs, complicating the extraction of relevant insights and the production of coherent outputs. To address this, we propose Struct-X, a novel framework that operates through five key phases: "*read-model-fill-reflect-reason*" efficiently enabling LLMs to utilize structured data. It begins by encoding structured data into a topological space using graph embeddings, followed by filling in missing entity information with knowledge retrieval modules, and filtering out irrelevant tokens via a self-supervised module. The final phase involves constructing a topological network with selected tokens to further reduce the total token length for more effective LLM inference. Additionally, Struct-X includes an Auxiliary Module trained to generate prompts, aiding LLMs in analyzing structured data. Extensive experiments on open-source benchmarks, including the knowledge graph question-answer task and the long document reading comprehension task, show that Struct-X notably improves LLM reasoning in complex structured input context. Finally, we deployed Struct-X in a real-world financial report analysis task, where it exhibits enhanced reasoning capabilities when applied to authentic scenario. The code has been undergoing open-source development to facilitate easy replication.[1].

CCS Concepts

• **Computing methodologies** → **Knowledge representation and reasoning**; **Information extraction**; • **Applied computing**;

Keywords

Structured processing, knowledge reasoning, information retrieval, self-supervised learning, large language models

ACM Reference Format:
Xiaoyu Tan, Haoyu Wang, Xihe Qiu, Leijun Cheng, Yuan Cheng, Wei Chu, Yinghui Xu, and Yuan Qi. 2025. Struct-X: Enhancing the Reasoning Capabilities of Large Language Models in Structured Data Scenarios. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3690624.3709381

1 Introduction

In recent years, significant advancements have been made in the field of large language models (LLMs), particularly in natural language understanding [7]. This progress has been largely driven by extensive pre-training on vast text corpora [8], which has enhanced their generation capabilities. These advancements are often viewed as critical steps towards the development of artificial general intelligence (AGI) [23]. During the deployment of LLMs as general-purpose assistants for a variety of real-world applications, it becomes necessary for LLMs to process multimodal inputs. Among these inputs, structured data, like structured knowledge graphs (KGs), is particularly important [26]. These graphs, with their rich repository of entity relationships and hierarchical knowledge, have the potential to significantly enhance the reasoning capabilities of LLMs, leading to more precise and reliable inferences on related tasks. However, in real-world applications, the effective utilization of structured knowledge in LLMs presents a significant challenge [22]. A common approach is to flatten the structured information into a lengthy text sequence before inputting it into LLMs [16]. However, **this method often introduces an excessive amount of task-irrelevant context. Excess information**

*Authors contributed equally to this research.
†Corresponding author
[1] https://ambityuki.github.io/StructX_KDD25

KDD '25, August 3–7, 2025, Toronto, ON, Canada
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1245-6/25/08
https://doi.org/10.1145/3690624.3709381

can overwhelm the models, thereby impairing inference efficiency and accuracy [10]. Additionally, it hinders the ability of LLMs to accurately comprehend and represent the complex knowledge embedded within structured data [48].

To address this issue, various approaches have been explored using knowledge graph triple conversion [47] and knowledge graph encodings [4, 14]. Additional related work is summarized in Appendix B. However, **these methods primarily concentrate on converting the structural data of knowledge graphs into different forms, which may overlook the challenge of processing redundant task-irrelevant information in structured data.** Moreover, **these approaches may face challenges in preserving the global topological structure of knowledge graphs**, a critical aspect related to the universal reasoning capabilities of LLMs.

Another significant challenge is the **high sparsity of these graphs [15], characterized by missing semantic connections between entities.** This sparsity presents a challenge for leveraging structural data in LLMs [9]. LLMs tend to prioritize explicit semantic connections presented in the context while overlooking implicit connections, which are crucial for enhancing inference performance. Although current research, such as [20] and [3], has been directed towards automatic knowledge completion and data augmentation to boost overall performance, these approaches tend to overlook the aforementioned challenges of redundancy and topological structure representation in utilizing structural data.

To address the bottlenecks mentioned above, we introduce **Struct-X**, a novel framework that enables LLMs to adapt to ***Struct***ured and redundant data scenarios with enhanced comple***X*** reasoning capabilities. This framework is centered around a workflow of "*read-model-fill-reflect-reason*". It first employs the transformation of structured data into a topological space through graph embeddings. Then, the augmentation of incomplete entity information is performed by knowledge retrieval modules. Subsequently, a self-retrieved generation module called Self-Reg is employed to eliminate irrelevant tokens. Finally, we employ a topological network to incorporate the chosen tokens and diminish the overall token length, thereby enhancing the efficacy of LLM reasoning. Furthermore, an Auxiliary Module is also designed in Struct-X, which adjusts prompts based on the loss, guiding the LLM generation.

We first assess the Struct-X using open-source benchmarks for knowledge graph QA and reading comprehension, which have highlighted its superior reasoning capabilities in handling structured data. Subsequently, we apply the Struct-X to perform real-world financial reports analysis task, where it further demonstrates its advanced reasoning skills with complex, structured financial data. We refer interested readers to Appendix A for more information about Struct-X's cases. **The code of Struct-X has also been open-sourced for easy replication**[1]**.** The main contributions of this paper include:

(1) We propose a novelty framework Struct-X that implements a process of "*read-model-fill-reflect-reason*" on structured data, enabling LLMs to perform effective complex reasoning over structured data.
(2) In Struct-X, We design a knowledge learning and filtering process to dynamically fill in structured knowledge gaps, coupled with a self-retrieved generation module called Self-Reg to filter the relevance of retrieved knowledge, retaining tokens with valuable information for effective reasoning.
(3) We also design an Auxiliary Module for generating coherent prompts and improving responses.
(4) Our proposed Struct-X has been evaluated on both open-source benchmarks and real-world scenarios to ensure its effectiveness across diverse structured data situations.

2 Preliminaries

The task of text generation in LLMs involves creating a sequence of output $y = [y_1, ..., y_T]$, where T represents the total number of tokens [31], based on a given input prompt x. This process is often modeled in an autoregressively manner, which estimates the likelihood of each token, where $y_{<t}$ represents the tokens that come before the current sequence $[y_1, ..., y_{t-1}]$ [46]. Enhancements to this process can be made by incorporating relevant information from external documents D into the input, thereby refining the model's predictions [12].

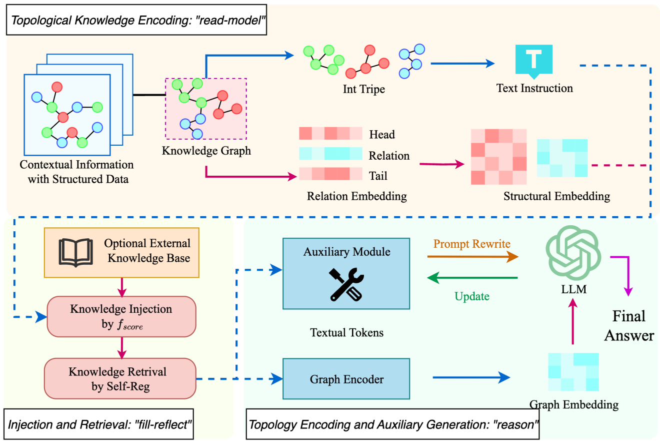

Topological Knowledge Encoding: "read-model"
Contextual Information with Structured Data
Knowledge Graph
Int Tripe
Text Instruction
Head
Relation
Tail
Relation Embedding
Structural Embedding
Optional External Knowledge Base
Knowledge Injection by f_{score}
Knowledge Retrival by Self-Reg
Injection and Retrieval: "fill-reflect"
Auxiliary Module
Prompt Rewrite
Update
LLM
Final Answer
Textual Tokens
Graph Encoder
Graph Embedding
Topology Encoding and Auxiliary Generation: "reason"

Figure 1: Struct-X framework comprises several key components to implement "*read-model-fill-reflect-reason*" process: topological knowledge encoding, knowledge injection and retrieval, graph topology encoding, and auxiliary module.

Moreover, we can develop a novel decoding strategy that produces critique tokens C alongside the main text output. These tokens are generated at each step and are designed to enable the LLMs to self-evaluate aspects such as relevance, factuality, and completeness of the generated content:

$$p(y, C|x) = \prod_{t=1}^{T} p(y_t, C_t|x, y_{<t}, C_{<t}), \tag{1}$$

where the critique token C_t depends on all preceding text and critiques. Here, we define four types of critique tokens which are demonstrated in the Table 1 to construct the whole Self-Reg module: IFReT - predicts if retrieval is needed, IFReL - assesses passage relevance, IFSuP - checks output is supported and IFUsE - decides whether it is useful. Readers may refer to Section 3.2.1 for an overview of the process for Self-Reg.

3 Methods

3.1 Topological Knowledge Encoding and Knowledge Injection

Here we first introduce the "*read-model*" process. Following the previous work [24], we transform the structured textual data into Knowledge Graphs (KGs). We start by processing the acquired KGs using a L-layers graph attention encoder (GAE) [38] to acquire the node feature embedding h_v, which can effectively represent both the semantic and structural information of the KGs. Then, the acquired h_v are then partially masked under a specific mask rate p_{mask}: $\tilde{h}_v = M_{p_{mask}}(h_v)$, to improve the overall training robustness and capability of processing missing information.

Then, we introduce the "*fill*" process. To address the gaps in entity information within the structured knowledge graph, we have developed a knowledge learning module to retrieve supplement facts from the knowledge base to enhance the masked node embeddings $\tilde{h}_v$ [44]. More specifically, for each masked node, we calculate a similarity score between its embedding $\tilde{h}_v$ and all tail entities t_v that are part of the entity set E. This is achieved by using the scoring function f_{score}, which can be represented as:

$$s(t_v) = f_{score}(\tilde{h}_v, t_v). \tag{2}$$

This process enables us to efficiently fill in the missing information in the knowledge graph from the masking operation $M_{p_{mask}}$.

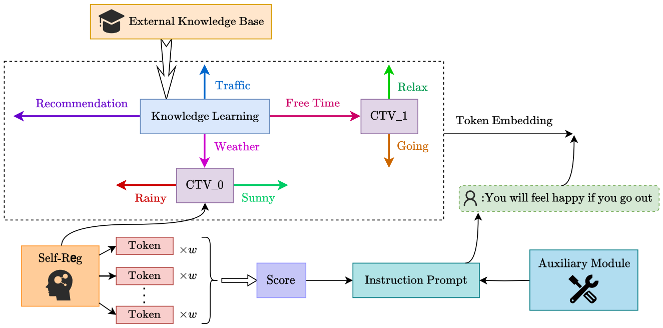


Figure 2: Knowledge injection and retrieval process in Struct-X with an example input: "Where can we go this weekend?".

The scoring function evaluates both feature and topological similarities within the graph which is shown in Figure 2. Besides the knowledge inside the graph, we can also perform a knowledge retrieval from the provided knowledge graph database to augment the generation. Both selection processes filter top K entities based on the highest scores and retrieve the related facts t from the knowledge base and t_v from the E. To incorporate these facts into the node embeddings, a relation-aware aggregation function f_{agg} is used. This function accumulates relevant knowledge for each node, using a score threshold τ to filter out irrelevant facts [45]. The aggregation function adeptly manages various relations in structured knowledge by considering information from retrieved triples in a relation-aware manner. Additionally, before being input into the Transformer encoder, one linear layer o concatenates and processes embeddings from all L GAE layers.

$$\begin{aligned} \bar{h}_v &= f_{agg}(\{\tilde{h}_v\} \cup \{(t, t_v) | s(t) > \tau, s(t_v) > \tau\}), \\ e_v &= o([h_v^{(1)}, ..., h_v^{(L)}, \bar{h}_v]). \end{aligned} \tag{3}$$

This process retains rich multi-scale structural and semantic features. The output e_v is flattened and prepared into sequences to replace token embeddings for the subsequent Transformer encoder input, as suggested by [36]. The refined node embeddings $\bar{h}_v$, enriched with retrieved entity information, supply additional knowledge for the reasoning in the downstream process of LLMs.

3.2 Knowledge and Information Retrieval

Here we introduce the "*reflect*" process. Although the f_{score} in the "*fill*" process can ensure the high similar entities to supplement the embedding information, it is important to filter out potential irrelevant information that may influence the subsequent generation process.

To filter and verify the relevance of retrieved knowledge, we design a self-retrieved generation process Self-Reg$_\psi(k)$ parameterized by ψ that takes as input the retrieved knowledge sequences k and outputs a filtered subset $\hat{k}$ containing only the most valuable tokens [29]. Specifically, Self-Reg$_\psi$ first encodes the knowledge sequence $k = (x_1, x_2, \ldots, x_N)$ using a Transformer encoder to obtain representations $h_i = f_{enc}(x_i)$. Then, we compute an importance score for each token $s_i = \sigma(g_{score}(h_i))$, where g_{score} is a scoring network and σ is a Sigmoid activation function. To train the scoring network in a self-supervised manner, we create corrupted knowledge sequences $\tilde{k}$ by randomly masking or shuffling some tokens. A contrastive loss is implemented to assign higher scores s_i to tokens from the original k versus corrupted $\tilde{k}$:

$$\mathcal{L}_{contrast} = \sum_i \max(0, s_i - \tilde{s_i} + \Delta), \tag{4}$$

where Δ is a margin hyperparameter. This drives the model to identify the most valuable knowledge. Finally, we filter the sequence by discarding tokens scoring below a threshold of τ to retain only the most relevant phrases, significantly reducing the learning burden when provided as supplements to the LLMs.

$$\hat{k} = \{x_i | s_i > \tau\}. \tag{5}$$

The filtered relevant knowledge $\hat{k}$ provides targeted assistance to improve reasoning without overwhelming the LLMs with extraneous and irrelevant information. Further details on knowledge filtering effects can be found in Section 5.0.5.

3.2.1 *Self-Reg Module.*

Here we introduce the submodules used in the Self-Reg process to verify the supplement information, which is also shown in Table 1.

IFReT Module This module decides if passage retrieval is needed using a scoring function:

$$\boxed{\text{IFReT}}(x) = f_\phi(x), \tag{6}$$

where x is the input text, and f_ϕ outputs a binary decision on whether to activate retrieval given x, parameterized by ϕ. For example, if the input is x: "Tell me more about Van Gogh's paintings", the

Table 1: Self-supervised Module related parameters for selective knowledge retrieval and response correlation verification

Type	Inputs	Outputs	Descriptions
IFReT	*query, context*	$\{Yes, Continue, No\}$	Determines if additional information is needed or if the response should continue
IFReL	*query, evidence*	$\{Relevant, NotRelevant\}$	Assesses relevance of evidence to the query
IFSuP	*query, evidence, response*	$\{FullySupported, PartiallySupported, NotSupported\}$	Evaluates how well the response is supported by the evidence
IFUsE	*query, response*	$\{5, 4, 3, 2, 1\}$	Rates the usefulness of the response in addressing the query

module may predict $\boxed{\text{IFReT}}(x) = 1$, indicating that retrieval would be useful to supplement details about Van Gogh's works.

IFReL Module This module scores the relevance of a retrieved passage p using:

$$s_{rel} = g_\theta(x, p) \cdot \sigma(\boxed{\text{IFReL}}(x, p)), \tag{7}$$

where g_θ produces a relevance score between input text x and passage p, modulated by the $\boxed{\text{IFReL}}(x, p)$ gate value passed through a sigmoid σ. For instance, if a retrieved passage discusses Surrealism instead of Van Gogh, the model can set a lower $\boxed{\text{IFReL}}(x, p)$ score to downweight it.

IFSuP Module This evaluates the factual consistency between response y and passage p:

$$s_{con} = h_\psi(y, p) \odot \sigma(\boxed{\text{IFSuP}}(y, p)), \tag{8}$$

where $\odot$ is element-wise production, h_ψ calculates consistency between y and p, controlled via $\boxed{\text{IFSuP}}$. This helps verify if details in y like dates or places align with the evidence in p.

IFUsE Module This directly outputs a usefulness score u between input x and response y: $u = \boxed{\text{IFUsE}}(x, y)$. For example, u may be lower if y fails to answer the query in x about Van Gogh's paintings. The modules apply self-supervision for relevance, coherence, and consistency.

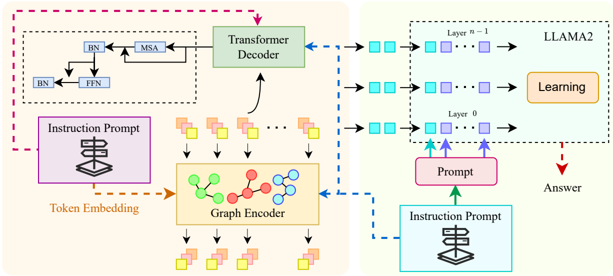


Figure 3: Interaction between graph and LLM in Struct-X. The encoder refines node embeddings via cross-layer passing. The condensed embeddings are provided as supplements.

The $\boxed{\text{IFReT}}(x)$ module for selective passage retrieval plays a key role in improving accuracy by retrieving evidence only when needed, avoiding unnecessary information. For instance, in closed-domain QA, Struct-X achieves higher recall by learning a tight retrieval threshold via $\boxed{\text{IFReT}}(x)$, while open-ended generation benefits from more selective retrieval. Furthermore, the $\boxed{\text{IFReL}}(x,p)$ module filters out lower-quality passages that are less relevant, as quantified by the relevance score s_{rel} in Eq.3 and calibrated by $\boxed{\text{IFReL}}$ gates. This enhances the contextual signals passed to the language model. The $\boxed{\text{IFSuP}}(y,p)$ and $\boxed{\text{IFUsE}}(x,y)$ critiques help further verify passage-response consistency and overall utility, ensuring higher quality outputs.

3.3 Graph Topology Encoder

Here we perform the "*reason*" phase. To capture semantic and structural interactions between entities within the KGs, we use a specialized graph encoder in Figure 3, denoted as $E_\theta(G)$, which is parameterized by θ. This KG is represented as $G = (V, E)$, where V is the set of node entities and E is the set of relation edges [17]. For each entity node v_i in V, we first derive its initial feature representation $h_{v_i}^{(0)}$, which is a vector in a high-dimensional space.

The graph encoder works through a series of L layers, each layer enhancing the node representations through message passing. This process can be described as:

$$h_{v_i}^{(l+1)} = f_\theta\left(\left\{h_{v_j}^{(l)} : v_j \in \mathcal{N}(v_i)\right\}\right), \quad \forall v_i \in V, \tag{9}$$

where $\mathcal{N}(v_i)$ refers to the neighboring nodes of v_i, and $f_\theta(\cdot)$ is a function that aggregates information from these neighbors to update the node's embedding.

To focus on the most relevant semantic connections, we use a graph self-attention layer within $f_\theta(\cdot)$. This layer calculates attention weights as follows $a_{ij} = \frac{\exp(\langle q_i, k_j \rangle)}{\sum_{v_k \in \mathcal{N}(v_i)} \exp(\langle q_i, k_k \rangle)}$, where q_i and k_j are derived from the embeddings of the nodes. This method allows the model to selectively emphasize the most informative signals from neighboring nodes [5].

After processing through L layers, we obtain refined node embeddings $z_{v_i} = h_{v_i}^{(L)}$, which encapsulate both semantic and structural information of the graph. To make these embeddings more manageable for downstream tasks, we compress them through a trainable down-projection layer:

$$e_{v_i} = \mathbf{W}_d z_{v_i}, \quad \mathbf{W}_d \in \mathbb{R}^{d_h \times d_e}, d_e < d_z. \tag{10}$$

This step reduces the dimensionality of the embeddings to d_e, which is smaller than the original d_z. The resulting condensed embeddings e_{v_i} still retain crucial token-level interactions but are more concise, making them better suited for training models for specific tasks. This approach ensures that while the size of the input sequence is significantly reduced, the essential semantic and structural features of the knowledge graph are preserved for subsequent reasoning.

3.4 Auxiliary Module

To further guide the LLMs in effectively reasoning over the structured input with a knowledge graph, we have developed an Auxiliary Module. This module is designed to create dynamic prompts that enhance the coherence of answers generated by LLMs. It functions by analyzing the LLM's predicted answer, denoted as $\hat{y}$, along with the current loss, L. Based on these inputs, it generates a refined prompt, p', which is then used for a new round of inference. We use the pre-trained Bert model (i.e., bert-base-NER) [6] to construct this Auxiliary Module, symbolized as G and parameterized by θ_g. This generator crafts the prompt text, taking into account the input values $p' = G(L, \hat{y}; \theta_g)$. The generator is trained jointly with the overall system using policy gradient methods to maximize the expected reward R of producing coherent answers: $J(\theta_g) = \mathbb{E}p' \sim G[R(p')]$, $\nabla\theta_g J(\theta_g) = \mathbb{E}p' \sim G[\nabla\theta_g \log G(p'|L, \hat{y}; \theta_g) R(p')]$.

This reward function is designed to encourage the LLMs to generate responses that are not only fluent but also logically consistent, particularly when using the updated prompt. This feature enables the module to dynamically adjust prompts based on the current performance, thereby offering new approaches to improve the quality of answers. Throughout the training process, the module progressively learns to produce more effective prompts, leading to enhanced accuracy and coherence in the LLM's reasoning. For more detailed experimental testing and analysis, please refer to Section 4.3.

4 Experiment

4.1 Datasets and Tasks

We assess the performance of our proposed Struct-X framework on four open-source benchmark datasets designed for knowledge graph reasoning and multi-hop reasoning abilities on graphs.

Task1:WebQSP contains 4,737 QA pairs where the questions require logical reasoning over a knowledge graph derived from Wikipedia to infer the correct answer. The knowledge graph consists of 5,719 entities and 2,150 relations.

Task2:MetaQA comprises a set of more complex compositional questions constructed from an underlying knowledge graph with a vocabulary of 300 entities and 100 relations. It has a total of 1,200 unique questions that test the multi-hop, logical, and comparative reasoning abilities of models.

Task3:Family Tree Age Consider a family tree $G = (V, E)$, where each individual v_i in V is associated with a description d_i specifying their age. The objective of this task is to identify the triplet comprising an individual, one of their grandparents, and a grand-uncle/grand-aunt by marriage that collectively has the highest cumulative age.

Task4:Travel Route Optimization Let $G = (V, E)$ be a graph representing connected cities, where each city v_i in V has a description d_i with the travel toll or tax. The LLM must plan the route from a source to a destination city that minimizes the total toll paid.

For all datasets, we incorporate the encoded graph representations into Llama2, which has been pre-trained on BookCorpus and English Wikipedia. Section 7 is a case analysis of the experimental results on the datasets.

4.2 Implementation Details

Baseline

- **Embedding-based Model**: We compare against representative embedding models for knowledge graph reasoning including TransE [2], DistMult [40], EmbedKGQA [27], ComplEx [34], and RotatE [30].
- **Open-source LLM**: We evaluate reasoning capabilities of widely-used pre-trained language models accessible through open APIs, including Llama2 [7B & 13B] [32] and Alpaca [7B & 13B] [42] which are openly available LLMs up to 13 billion parameters.
- **LLM-based Fine-tuning**: To assess the performance of LM fine-tuning approaches, we include as baselines KG-LlaMA and KG-Alpaca [43], KG-BERT [41], PKGC [20] and vanilla IT [47] which incorporate techniques to enhance LMs using annotated KG datasets or self-supervision.

Our implementation is in PyTorch and we run experiments on NVIDIA A100 GPUs. The hyperparameters in Table 2 provide concrete configuration details for Struct-X when evaluated on the four benchmark datasets.

Table 2: Hyperparameters of Struct-X for the four datasets

Hyperparameter	WebQSP	MetaQA	Family Tree	Travel Route
Graph Encoder Layers	4	4	4	4
Graph Encoder Dimensions	512	512	512	512
Encoder Heads	8	8	8	8
Node Masking Rate	0.4	0.4	0.3	0.3
Auxiliary Dimensions	512	512	256	256
Prefix	5	5	5	5
Generator Layers	2	2	4	4
MLM Probability	0.2	0.3	0.15	0.2
Tunable parameters	0.0933 B	0.0933 B	0.0933 B	0.0933 B
Batch Size	32	32	16	16
Dropout	0	0	0	0
Prefix	5	5	5	5
Batch Size	32	32	16	16
Learning Rate	5×10^{-5}	5×10^{-5}	1e-4	1e-4
Training Epochs	15	20	15	10
Warmup epochs	1	1	1	1
Weight decay	0.01	0.01	0.01	0.01

4.3 Main Results

The results presented in Table 3 indicate that Struct-X consistently outperforms existing baseline methods across various datasets. Specifically, in the WebQSP benchmark, Struct-X achieves an accuracy of 75.13%, which is 2.65% higher than the previously best-performing method, KoPA. Additionally, Struct-X shows modest improvements in precision and recall, with increases of 9.51% and 10.96%, respectively, compared to Vanilla IT. In the more challenging MetaQA dataset, Struct-X's performance is notably better, surpassing the state-of-the-art accuracy scores by 1.84% and achieving a 1.68% higher precision. Furthermore, Struct-X demonstrates significant advancements in specialized tasks such as Family Tree and Travel Route, where it exceeds the top baseline results by 3.36% and 5.34% in accuracy, respectively.

Compared to embedding models such as TransE, DistMult, and EmbedKGQA, Struct-X also shows promising improvements in reasoning abilities by integrating both semantic and topological structures of knowledge graphs. For instance, against RotatE's accuracy of 74.55% on the WebQSP dataset, Struct-X achieves higher performance with a 75.13% accuracy, an increase of 0.58%. The difference is slightly more pronounced on the MetaQA dataset, where Struct-X exceeds RotatE's score of 78.19% by 1.44% in accuracy. In scenarios requiring complex reasoning inferences, Struct-X demonstrates enhanced capabilities, outperforming peak embedding model accuracy by a notable margin of 16.74% in Task4.

Table 3: Performance comparison across different datasets and Tasks

Model/Datasets	WebQSP				MetaQA				Family Tree				Travel Route			
	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1
Embedding-based Model																
TransE	67.32	68.41	65.47	66.91	74.63	74.44	75.01	74.72	73.45	71.83	66.92	63.84	62.27	60.45	65.28	62.56
DistMult	69.21	69.91	69.08	69.49	69.34	72.32	61.93	62.83	60.93	61.47	60.08	60.62	61.74	60.36	64.19	62.05
EmbedKGQA	63.26	72.38	74.33	73.35	70.17	70.38	69.57	69.97	65.62	64.31	68.94	65.92	64.95	63.01	68.64	65.37
ComplEx	67.36	68.21	66.27	65.64	72.86	71.64	71.18	69.53	59.83	58.45	62.74	60.27	57.45	55.92	60.93	57.84
RotatE	74.55	72.68	76.93	74.77	78.19	78.16	**78.22**	**77.19**	67.63	66.28	65.79	64.37	62.64	63.01	61.83	62.27
Open-source LLM																
Llama2$_{7B}$	27.13	29.17	25.23	27.36	26.45	25.73	28.82	26.94	32.81	34.73	30.16	32.26	25.92	27.84	23.68	25.47
Alpaca$_{7B}$	33.56	31.83	36.29	33.72	22.74	25.93	19.82	21.45	28.36	26.92	31.74	28.91	39.45	41.27	37.63	39.18
Llama2$_{13B}$	29.83	28.73	31.91	39.95	35.92	36.19	36.84	35.91	31.45	33.82	39.27	31.28	36.37	38.62	34.93	36.45
Alpaca$_{13B}$	38.37	40.92	36.74	38.45	42.83	33.62	31.74	32.56	40.83	32.76	38.84	40.37	37.92	35.74	40.83	37.94
ChatGPT	41.27	49.84	44.92	41.72	38.74	37.56	40.83	38.94	35.92	33.84	38.62	35.83	34.83	36.73	42.15	44.71
LLM-based Fine-tuning																
KG-LLaMA	42.45	43.28	40.39	41.37	46.28	45.94	47.36	46.47	49.74	48.36	42.64	40.27	48.56	50.28	45.74	47.83
KG-Alpaca	48.92	47.74	51.83	49.45	43.56	42.41	46.28	43.94	45.92	46.84	44.56	45.62	42.91	50.74	47.28	43.45
KG-BERT	56.28	55.94	57.36	56.47	70.92	69.74	63.82	61.45	63.27	61.19	68.29	64.23	74.32	75.61	74.45	72.37
PKGC	64.56	67.44	62.64	64.10	77.79	76.92	74.27	73.49	79.32	78.41	75.36	74.91	76.35	75.42	74.37	76.91
Vanilla IT	65.62	69.86	66.29	65.54	78.28	74.31	72.98	73.62	71.57	70.58	69.63	70.05	65.94	67.32	63.69	64.96
KoPA	72.48	72.82	71.64	71.52	75.82	74.69	73.95	71.58	73.56	72.18	72.08	71.53	73.13	75.05	71.12	71.24
Ours(Struct-X)	**75.13**	**73.40**	**77.25**	**75.29**	**79.63**	**78.27**	77.53	76.61	**82.68**	**82.95**	**79.34**	**78.92**	**81.69**	**81.53**	**78.62**	**78.04**

The results also show that Struct-X can enhance the capabilities of the Llama2, which itself achieves a 27.13% accuracy on the WebQSP benchmark. This enhancement is achieved through masking graph embeddings and using topology matching to retrieve relevant facts, thus addressing the gaps in factual knowledge that Llama2 requires. By overcoming these deficiencies in LLMs, Struct-X significantly improves performance, increasing accuracy by 47.4%. This indicates the effectiveness of structured augmentation, which is not present in the Llama2. Further, Struct-X filters out less important tokens using the Self-Reg module, ensuring focus on the most relevant information. In comparison to previous methods like KG-BERT fine-tuning, Struct-X offers essential enhancements, particularly in complex reasoning tasks, as evidenced by increases of up to 10.24% in accuracy and 5.61% in recall.

Based on the experimental results, the "*reflect*" process also plays a crucial role in enhancing reasoning capabilities. This process involves the IFReT (x), which selectively gathers evidence as needed, and the IFReL (x, p), which filters less relevant passages using relevance scores from Eq.3 to improve context for LLMs. Additionally, the IFSuP (y, p) and IFUsE (x, y) ensure passage-response consistency and assess overall utility, contributing to higher quality results.

Figure 4 illustrates performance on four reasoning tasks under variants of the Struct-X framework with different components disabled. For multi-hop QA (Fig 4a), removing the knowledge retrieval module causes an evident drop of 8.2% in accuracy. This validates its role in traversing distant nodes by filling gaps. Omitting the encoding also degrades topological reasoning, seen in the 4.3% lower location QA (Fig 4b) accuracy without the graph encoder. For complex inferences (Fig 4c), the Auxiliary Module contributes to steering coherent deductions, reflected by 6.5% weaker decomposition scores when absent. Finally, inconsistent responses increase by 4.9% (Fig 4d) after disabling relevance filtering, confirming the module's benefits for factual verification.

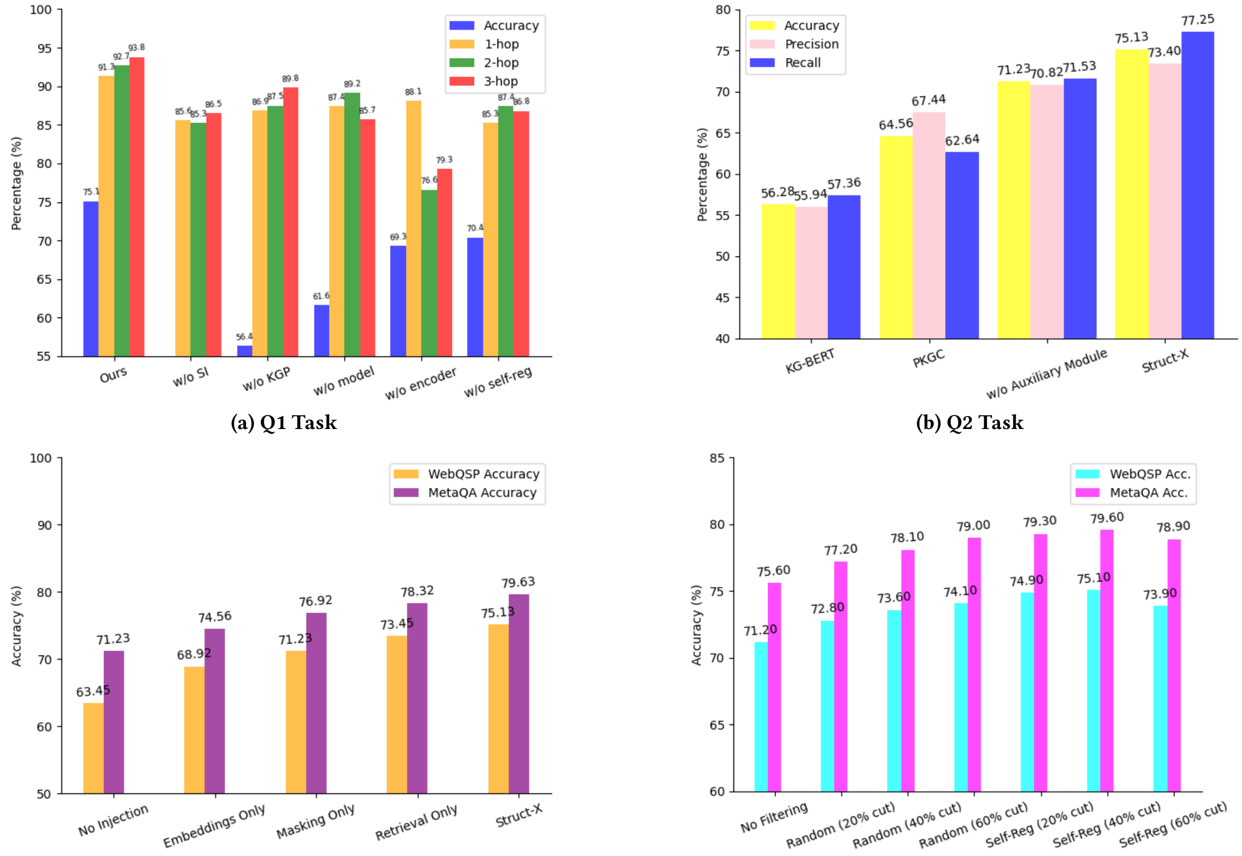
Table 4: Performance evaluation and comparison across different functional modules on reasoning tasks.

Methods	WQSP	QA 1hop	QA 2hop	QA 3hop
KV-Mem	48.3	84.3	74.5	46.2
GraftNet	53.1	84.1	72.3	62.3
EmbedKGQA	63.3	85.3	74.1	79.8
NSM	69.6	84.6	89.1	88.7
UniKGQA	71.6	86.8	82.8	92.3
StructGPT	64.8	87.4	87.1	92.9
Ours	**75.1**	**91.3**	**92.7**	**93.8**
w/o SI (struct inference)	54.2	85.6	85.3	86.5
w/o KGP (KG perform better)	56.4	86.9	87.5	89.8
w/o model (graph study)	61.6	87.4	89.2	85.7
w/o encoder (location)	69.3	88.1	76.6	79.3
w/o self-reg (correction)	70.4	85.3	87.4	86.8

Across diverse reasoning dimensions, targeted ablation of key components consistently reveals measurable impacts validating their individual contributions. The knowledge retrieval module facilitates connecting facts between entities, the graph encoder grounds topological representations, the Auxiliary Module directs logical inferences, and relevance filtering retains consistency. As a novel framework, Struct-X combines complementary mechanisms for enhancing language models' structured reasoning capacities. The ablation analyses quantify these modular effects through performance differentials on specialized probing tasks.

5 Ablation Study

5.0.1 Q1: Different functional modules. Table 4 shows that each component of Struct-X plays a crucial role in enhancing various reasoning capabilities. For 1-hop single fact questions, while all versions of Struct-X are effective, the complete model excels with a 91.3% accuracy due to its ability to perform combinatorial



(a) Q1 Task

(b) Q2 Task

(c) Q3 Task

(d) Q4 Task

Figure 4: The results of four tasks in experiments section

reasoning using multi-head attention. This is key for interpreting semantic connections. In 2-hop and 3-hop multi-step reasoning, the absence of knowledge retrieval and injection modules results in a significant performance drop, with decreases of 7.4% and 7.3% respectively. However, the full Struct-X model, utilizing these modules, reaches 92.7% and 93.8% accuracy by effectively traversing distant nodes. The graph topology encoder also proves vital; its omission leads to a 5.8% decline in location-based reasoning, highlighting its importance in connecting nodes and facilitating spatial/hierarchical reasoning through message passing. Furthermore, the lower accuracy without the Auxiliary Module underlines its utility in guiding coherent inference across multiple steps.

5.0.2 *Q2: Filtering and reflection mechanism.*

Table 5 compares reasoning performance with the following variants: $StructX_{NoFiltering}$: Directly injects all the retrieved knowledge without need filtering. $StructX_{RandomFiltering}$: Randomly removes of retrieved tokens. $StructX_{RegFiltering}$: Uses the proposed Self-Reg module to score and filter tokens.

Across WebQSP and MetaQA datasets, incorporating filtering mechanisms leads to consistent gains over no filtering baselines. Randomly removing tokens brings minor improvements, showing that some knowledge reduction is beneficial. However, learned filtering with Self-Reg leads to more substantial gains. Comparing different Self-Reg cutting ratios, 40% filtering seems to achieve the optimal trade-off, maximizing accuracy and recall. More aggressive 60% cutting starts to degrade performance likely due to removing pertinent facts. On the other hand, light 20% filtering retains more distracting information. By balancing knowledge breadth and depth, 40% Self-Reg filtering enhances language model inference without overwhelming models. By scoring and removing extraneous tokens based on contextual representations, Self-Reg retains the essence to augment language models without diverting attention.

5.0.3 *Q3:Knowledge injection variants.*

To validate the contributions of different components of our knowledge injection mechanism, we conduct an ablation study with the following variants: $StructX_{NoInjection}$: The base LLM (i.e., Llama2) without any graph representation injection. $StructX_{EmbeddingsOnly}$: Encoded graph embeddings are directly injected without any masking or knowledge retrieval. $StructX_{MaskingOnly}$: Graph embeddings are masked but missing facts are not filled via retrieval. $StructX_{RetrievalOnly}$: Masked embeddings are completed with the knowledge retrieval module but without graph encoding. We compare reasoning performance on WebQSP and MetaQA benchmarks against these reduced injection variants. The results in Table 6 demonstrate clear improvements from collectively incorporating knowledge injection components compared to ablated variants. The full Struct-X model with topological encoding, masking, and retrieval achieves 1.68% and 1.31% higher accuracy over best partial variant on WebQSP

Table 5: Analysis of using different token filtering ratios in Self-Reg module

Model Variant	WebQSP			MetaQA		
	Acc. (%)	Rec. (%)	F1 (%)	Acc. (%)	Rec. (%)	F1 (%)
No Filtering	71.2 ± 1.41	68.4 ± 1.59	69.8 ± 1.47	75.6 ± 1.19	72.3 ± 1.38	73.9 ± 1.26
Random (20% cut)	72.8 ± 1.28	70.1 ± 1.44	71.4 ± 1.33	77.2 ± 1.07	74.2 ± 1.24	75.6 ± 1.13
Random (40% cut)	73.6 ± 1.21	71.3 ± 1.36	72.6 ± 1.26	78.1 ± 1.01	75.4 ± 1.17	74.7 ± 1.08
Random (60% cut)	74.1 ± 1.16	72.8 ± 1.30	73.3 ± 1.21	79.0 ± 0.95	76.8 ± 1.11	75.9 ± 1.03
Self-Reg (20% cut)	74.9 ± 1.09	73.2 ± 1.23	74.0 ± 1.14	79.3 ± 0.92	78.2 ± 1.06	76.1 ± 0.99
Self-Reg (40% cut)	**75.1 ± 1.07**	**73.4 ± 1.21**	**75.3 ± 1.12**	**79.6 ± 0.90**	**78.3 ± 1.04**	**76.6 ± 0.96**
Self-Reg (60% cut)	73.9 ± 1.19	72.7 ± 1.34	73.2 ± 1.24	78.9 ± 0.97	77.8 ± 1.12	76.4 ± 1.03

and MetaQA respectively. This confirms each mechanism provides unique benefits - topological encoding better retain intricate connections, masking identifies missing facts, and retrieval fills knowledge gaps. The experiment proves dynamic masking and retrieval to address inherent incompleteness in structured data are most impactful. Variants without these processes show worse performance as they fail to overcome language models' factual deficiencies.

Table 6: Advantages of dynamic factual injection and retrieval over individual knowledge supplementation variants

Model Variant	WebQSP Accuracy (%)	MetaQA Accuracy (%)
No Injection	63.45 ± 1.72	71.23 ± 1.43
Embeddings Only	68.92 ± 1.37	74.56 ± 1.21
Masking Only	71.23 ± 1.19	76.92 ± 1.08
Retrieval Only	73.45 ± 1.04	78.32 ± 0.92
Struct-X	**75.13 ± 0.98**	**79.63 ± 0.84**

5.0.4 Q4:Performance comparison on different scenarios. The experimental results demonstrate the effectiveness of our proposed Struct-X across both BioASQ and MedQA tasks in Table 7. Looking at the performance progression, we observe significant improvements over baseline approaches. Specifically, Struct-X achieves 79.8% accuracy on BioASQ and 77.5% on MedQA, representing notable improvements of 11.6% and 12.2% respectively compared to the KG-BERT baseline. The consistent performance gains across both tasks suggest that our model's structural learning capabilities are robust and generalizable. Compared to the strong PKGC baseline, Struct-X shows improvements of 2.6% on BioASQ and 2.4% on MedQA, indicating that our architectural enhancements effectively capture complex relationships in the data. The performance gap between EmbedKGQA (72.5% on BioASQ, 70.2% on MedQA) and RotatE (75.9% on BioASQ, 73.6% on MedQA) highlights the importance of rotational embeddings for capturing fine-grained semantic relationships. Furthermore, the parallel improvements in F1 scores (79.1% for BioASQ and 76.7% for MedQA) demonstrate that Struct-X maintains balanced precision and recall, suggesting it's not just making more accurate predictions but also providing more comprehensive coverage of the answer space.

Table 7: Comparison on BioASQ and MedQA tasks

Model	BioASQ		MedQA	
	Accuracy (%)	F1 Score (%)	Accuracy (%)	F1 Score (%)
KG-BERT	68.2	67.6	65.3	64.8
EmbedKGQA	72.5	71.8	70.2	69.5
RotatE	75.9	75.2	73.6	72.9
PKGC	77.2	76.5	75.1	74.3
Struct-X (Ours)	**79.8**	**79.1**	**77.5**	**76.7**

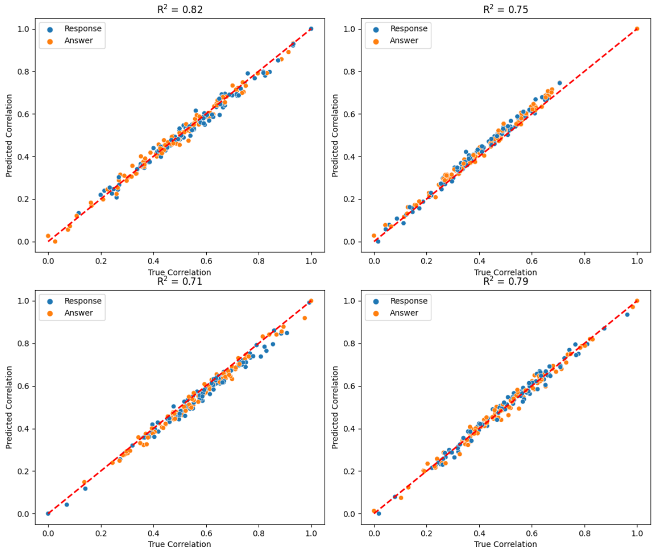


Figure 5: Visualization of performance of SelfReg module.

5.0.5 Q5:Knowledge Filtering Impact on QA. Figure 5 shows the performance of the model before and after applying different levels of knowledge filtering in question-answering comprehension tasks. The filtering ratio varies between 0 and 60%, and the improvement in fit between the response of the large language model and the real-world answer is used as the criterion for determining the effectiveness. Firstly, when there is no filtering (0%), the fitting degree R^2 is around 0.7. This is the original level when injecting all knowledge. Subsequently, we observed that as the filtering ratio increased, the fitting degree R^2 showed a trend of first increasing and then decreasing. When filtering out about 40% of low correlation knowledge, the model accuracy reaches a peak of around 0.82. This indicates that through algorithms such as Self Reg, the model has learned to recognize the most critical knowledge for the current question and answer. Overfiltering knowledge actually makes the model unable to learn comprehensively. However, continuing to increase the filtration ratio to 40-60% will result in a reversal and decline in the fit. The model has lost some useful knowledge, and the contextual information is insufficient for the model to make accurate inferences. Therefore, we validated and demonstrated that appropriate knowledge filtering can improve

Table 8: Performance in Financial Report Analysis

Model	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)	Avg. Response Time (s)
Llama2-7B	68.4 ± 0.7	69.7 ± 0.8	67.9 ± 0.9	68.8 ± 0.8	1.94 ± 0.12
Alpaca-7B	69.1 ± 0.8	70.3 ± 0.7	68.5 ± 0.8	69.4 ± 0.7	2.04 ± 0.15
Llama2-13B	73.8 ± 0.6	75.2 ± 0.7	74.6 ± 0.8	74.9 ± 0.7	1.66 ± 0.10
Alpaca-13B	74.2 ± 0.7	75.7 ± 0.6	74.9 ± 0.7	75.3 ± 0.6	1.76 ± 0.11
ChatGPT-3.5	76.5 ± 0.5	77.9 ± 0.6	78.2 ± 0.7	**78.0 ± 0.6**	1.38 ± 0.08
StructX (Ours)	**78.3 ± 0.4**	**80.1 ± 0.5**	**81.2 ± 0.6**	77.4 ± 0.5	**0.90 ± 0.05**

the effectiveness of question answering, but a balance needs to be found between denoising and preserving information.

6 Industrial Implementation in Finance

The ability to accurately interpret and analyze structured economic data in financial reports is essential for various operations, including finance, consulting, and strategic planning. In this study, we integrate our proposed Struct-X into our financial report analysis service to perform tabular financial question answering tasks. We compare the performance of our method against baseline models from online services and several popular open-source language models, including Llama2 (7B and 13B versions) [33], Alpaca (7B and 13B versions) [2], and ChatGPT [1]. The evaluation utilizes a dataset comprising Q2 financial reports from our database. The sample case of this task is demonstrated in the Appendix C.

As presented in Table 8, the Struct-X model has consistently delivered superior performance relative to competing models on key evaluation metrics. Specifically, Struct-X achieved an accuracy of 78.3%, a precision of 80.1%, and a recall of 81.2%. These figures represent an enhancement of 1.8 percentage points in accuracy, 2.2 points in precision, and 3.0 points in recall over the next best-performing baseline model.

When compared with smaller-scale models such as Llama2-7B and Alpaca-7B, the margin of improvement attributable to Struct-X becomes even more pronounced, approaching an around 10 percentage point increment across the aforementioned metrics. Furthermore, Struct-X has demonstrated a notable efficiency edge, with an average response time of 0.90 seconds. This response time is 35% quicker than that of ChatGPT-3.5 and exceeds the speed of 7B model variants by over 50%. In environments where financial analysis must be conducted in real-time, this expedited processing capability is invaluable, facilitating more prompt insights and enabling a more dynamic and responsive approach to decision-making in high-stakes, rapidly evolving business landscapes. Struct-X's combination of accuracy, speed, and comprehensive analysis capabilities positions it as a valuable tool for professionals dealing with complex financial and economic data in industrial settings.

7 Case Study

In this family tree example, Struct-X first encodes the hierarchical relations between members like "John is Nikhil's paternal grandfather" and "Meera is John's daughter" as structured knowledge. The connections allow tracing ancestry and descendent paths. Age details like "John was born in 1950" are also injected. With encoded topology and dates, Struct-X can traverse the tree to identify the oldest member, Shekhar and the youngest, Nikhil, before subtracting their years to find the 54-year difference. This showcases interpreting familial links and performing numerical reasoning over injected facts. By attending over neighborhood nodes, Struct-X focuses on the most relevant semantic connections. The topology learning distinguishes close and distant relatives to assess ages. Retrieved date details fill knowledge gaps for calculation.

The example proves Struct-X can encode intricate hierarchical structures and use encoded topology to trace relationships and inject valuable factual knowledge. By learning contextual representations and connections in structured data, Struct-X successfully interprets semantic links between entities and integrates supplementary date details for numerical reasoning over multiple inference steps. This supports complex reasoning across topological dimensions.

Family Relations
John is Nikhil's paternal grandfather. Nikhil's aunt Meera is John's daughter. John was born in 1950. Nikhil was born in 2000. Meera was born in 1975.
Raj is Nikhil's maternal grandfather. Raj has a brother named Shekhar. Raj was born in 1948. Shekhar was born in 1946.
Question: What is the age difference between the oldest and youngest family members described?
Response: Based on the family tree details provided:
- The oldest member is Shekhar, Nikhil's grand uncle, born in 1946.
- The youngest member is Nikhil, born in 2000.
- Their age difference is 1946 - 2000 = 54 years.

So the age difference between the oldest and youngest family members is **54 years**.
Correct Answer: 54 years old.

8 Conclusion

In this paper, we introduce Struct-X, a novel framework designed to enhance LLMs in complex reasoning tasks under structured data scenarios. Struct-X applies an efficient "*read-model-fill-reflect-reason*" process to perform reasoning over structured data. It is adept at learning graph embeddings that are sensitive to geometric contexts, capturing the content of entities as well as their topological relationships. This enables Struct-X to effectively fill missing facts about entities by matching similar topological features. Furthermore, it enhances the LLMs by distributing multi-scale features, which bolsters the representation of underlying connections that are not explicitly apparent in redundant contextual format. Struct-X excels in tasks such as knowledge graph-based QA tasks and reading comprehension, especially in scenarios that require multi-hop logical reasoning.

[2] https://crfm.stanford.edu/2023/03/13/alpaca.html

References

[1] Joshua Achiam, Shlomo Adler, Swetank Agarwal, Liza Ahmad, Ilya Akkaya, Francisco L. Aleman, and Brian McGrew. 2023. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774* (2023). https://arxiv.org/abs/2303.08774

[2] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *NIPS*. 2787–2795.

[3] Ziwei Chai. 2023. Graphllm: Boosting graph reasoning ability of large language model. *arXiv preprint arXiv:2310.05845* (2023).

[4] Zhikai Chen. 2023. Exploring the potential of large language models (llms) in learning on graphs. *arXiv preprint arXiv:2307.03393* (2023).

[5] G. Cui, J. Zhou, C. Yang, and Z. Liu. 2020. Adaptive graph encoder for attributed graph embedding. In *SIGKDD*. 976–985.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* (2018).

[7] A. Fan, B. Gokkaya, M. Harman, M. Lyubarskiy, S. Sengupta, S. Yoo, and J. M. Zhang. 2023. Large language models for software engineering: Survey and open problems. *arXiv preprint arXiv:2310.03533* (2023).

[8] L. Gao, Z. Dai, and et al. 2023. Rarr: Researching and revising what language models say, using language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 16477–16508.

[9] M. U. Hadi, R. Qureshi, A. Shah, M. Irfan, A. Zafar, M. B. Shaikh, and S. Mirjalili. 2023. A survey on large language models: Applications, challenges, limitations, and practical usage. *Authorea Preprints* (2023).

[10] J. Han, J. Kang Lu, Y. Xu, J. You, and B. Wu. 2024. Intelligent Practices of Large Language Models in Digital Government Services. *IEEE Access* (2024).

[11] Ayou Harnoune. 2021. BERT based clinical knowledge extraction for biomedical knowledge graph construction and analysis. *Computer Methods and Programs in Biomedicine Update* 1 (2021), 100042.

[12] Sebastian Hofstätter. 2023. Fid-light: Efficient and effective retrieval-augmented text generation. In *ACM SIGIR*.

[13] Or Honovich. 2022. TRUE: Re-evaluating factual consistency evaluation. *arXiv preprint arXiv:2204.04991* (2022).

[14] Jinhao Jiang. 2023. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645* (2023).

[15] A. Lazaridou, E. Gribovskaya, W. Stokowiec, and N. Grigorev. 2022. Internet-augmented language models through few-shot prompting for open-domain question answering. *arXiv preprint arXiv:2203.05115* (2022).

[16] Lei Li, Yongfeng Zhang, and Li Chen. 2023. Prompt distillation for efficient llm-based recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*.

[17] Yang Li. 2022. Dual encoder-based dynamic-channel graph convolutional network with edge enhancement for retinal vessel segmentation. *IEEE Transactions on Medical Imaging* 41, 8 (2022), 1975–1989.

[18] Weijie Liu. 2020. K-bert: Enabling language representation with knowledge graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34.

[19] Xiao Liu. 2022. Selfkg: Self-supervised entity alignment in knowledge graphs. In *Proceedings of the ACM Web Conference 2022*.

[20] Xin Lv, Yankai Lin, Yixin Cao, and et al. 2022. Do Pre-trained Models Benefit Knowledge Graph Completion? A Reliable Evaluation and a Reasonable Approach. In *ACL (Findings)*. Association for Computational Linguistics, 3570–3581.

[21] Shirui Pan. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering* (2024).

[22] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering* (2024).

[23] Jian Pei, Lei Deng, Shuai Song, and Ming Zhao. 2019. Towards artificial general intelligence with hybrid Tianjic chip architecture. *Nature* 572, 7767 (2019), 106–111.

[24] Chao Qu et al. 2020. Intention propagation for multi-agent reinforcement learning. (2020).

[25] Claude Roux. 2022. The Sydney declaration–Revisiting the essence of forensic science through its fundamental principles. *Forensic Science International* 332 (2022), 111182.

[26] Vincent Ryen, Ahmet Soylu, and Dumitru Roman. 2022. Building semantic knowledge graphs from (semi-) structured data: a review. *Future Internet* 14, 5 (2022), 129.

[27] Apoorv Saxena, Aditay Tripathi, and Partha P. Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *ACL 2020, Online, July 5-10, 2020*. 4498–4507.

[28] Koonal K. Shah. 2020. An exploration of methods for obtaining 0= dead anchors for latent scale EQ-5D-Y values. *The European Journal of Health Economics* 21 (2020), 1091–1103.

[29] Kurt Shuster. 2021. Retrieval augmentation reduces hallucination in conversation. *arXiv preprint arXiv:2104.07567* (2021).

[30] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *ICLR (Poster)*.

[31] Ruize Tang, Xianpei Han, Xia Jiang, and Xiaohua Hu. 2023. Does synthetic data generation of LLMs help clinical text mining? *arXiv preprint arXiv:2303.04360* (2023).

[32] Hugo Touvron et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[33] Hugo Touvron, Louis Martin, Kevin Stone, Pierre Albert, Amjad Almahairi, Yasin Babaei, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv preprint arXiv:2307.09288* (2023). https://arxiv.org/abs/2307.09288

[34] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex Embeddings for Simple Link Prediction. In *ICML*. JMLR.org, 2071–2080.

[35] Ke Tu. 2021. Conditional graph attention networks for distilling and refining knowledge graphs in recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*.

[36] Changgang Wang, Jun An, and Gang Mu. 2021. Power system network topology identification based on knowledge graph and graph neural network. *Frontiers in Energy Research* 8 (2021), 613331.

[37] Zonghan Wu. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* 32, 1 (2020), 4–24.

[38] H. Xu, W. Xia, Q. Gao, J. Han, and X. Gao. 2021. Graph embedding clustering: Graph attention auto-encoder with cluster-specificity distribution. *Neural Networks* 142 (2021), 221–230.

[39] Xiaoming Xue. 2020. Affine transformation-enhanced multifactorial optimization for heterogeneous problems. *IEEE Transactions on Cybernetics* 52, 7 (2020), 6217–6231.

[40] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *ICLR (Poster)*.

[41] Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. KG-BERT: BERT for Knowledge Graph Completion. *CoRR* abs/1909.03193 (2019).

[42] Liang Yao, Jiazhen Peng, Chengsheng Mao, and Yuan Luo. 2023. Exploring Large Language Models for Knowledge Graph Completion. *CoRR* abs/2308.13916 (2023).

[43] Liang Yao, Jiazhen Peng, Chengsheng Mao, and Yuan Luo. 2023. Exploring Large Language Models for Knowledge Graph Completion. *CoRR* abs/2308.13916 (2023).

[44] M. Yasunaga, A. Bosselut, and H. et al. Ren. 2022. Deep bidirectional language-knowledge graph pretraining. In *Advances in Neural Information Processing Systems*, Vol. 35. 37309–37323.

[45] D. Yu, C. Zhu, Y. Yang, and M. Zeng. 2022. Jaket: Joint pre-training of knowledge graph and language understanding. In *AAAI*, Vol. 36. 11630–11638.

[46] Yizhe Zhang. 2022. RetGen: A joint framework for retrieval and grounded text generation modeling. In *AAAI*, Vol. 36.

[47] Yichi Zhang. 2023. Making Large Language Models Perform Better in Knowledge Graph Completion. *arXiv preprint arXiv:2310.06671* (2023).

[48] Bin Zhou. 2023. Semantic-aware event link reasoning over industrial knowledge graph embedding time series data. *International Journal of Production Research* 61, 12 (2023), 4117–4134.

A StructX Interaction Examples

Instructions

Please indicate whether referring to external documents, improves the quality of the generated response. Please respond with either [Yes] or [No] and provide a brief explanation.

Instruction: Identify the shortest path between two nodes in this knowledge graph.
Need retrieval? [Yes]
Explanation: Struct-X can ingest the graph structure and topology to reason about paths. But retrieving additional facts on edge distances or weights can supplement its understanding for more accurate optimization.
Instruction: Determine which family tree node has the oldest relative based on date descriptions.
Need retrieval? [No]
Explanation: Struct-X encodes the hierarchical tree relations and date informations directly without needing external evidence. Retrieval may introduce unnecessary details.
Instruction: Analyze the impacts of this new tax policy based on economic concepts.
Need retrieval? [Yes]
Explanation: While Struct-X has some linguistic capabilities, retrieving domain knowledge on economics and regulations will improve understanding of entities and contextual impacts for better analysis.
Instruction: Summarize the key events in this 5-page history passage.
Need retrieval? [No]
Explanation: Struct-X is designed to ingest long document passages directly through encoders. No need for external info.
Instruction: Compare the costs of different flight options based on stop, mileage and fare data.
Need retrieval? [No]
Explanation: Struct-X can encode and reason over structured data tables natively. External retrieval of similar data is unneeded.

B Related Work

Prior efforts have explored various techniques to enhance language models with structured knowledge. Pan *et al.* [21] directly converted knowledge graph triples into textual statements as inputs. Liu *et al.* [18] embedded knowledge graphs into hidden layers of BERT models. Harnoune *et al.*[11] utilized memory modules to provide facts for multi-hop reasoning. Wu *et al.* [37] designed graph recurrent networks to propagate information over knowledge graphs. Roux *et al.*[25] employed reinforcement learning for path-finding in knowledge-augmented environments. Tu *et al.* [35] used graph convolutional networks to distill subgraph knowledge. Liu *et al.* [19] pruned knowledge graph inputs via self-supervised learning objectives. Honovich *et al.*[13] focused on improving factual consistency in conditional text generation. Shah *et al.* [28] iteratively expanded anchor sets for more comprehensive grounding. Xue *et al.* [39] optimized path selection on heterogeneous knowledge sources. While providing valuable techniques, existing methods lack consideration for efficiently filtering and condensing structured knowledge supplements tailored for language model integration. Our work aims to address these limitations through specialized contextual encoding, retrieval, and distillation components in the proposed Struct-X framework.

Algorithm 1 Topology Learning and Training

```
Input: knowledge graph $G = (V, E)$, LLM $M_\theta$, Auxiliary Module $A_\phi$
Encode $G$ into latent embeddings $Z_V$
Mask node embeddings at rate $p_{\text{mask}}$ as $\tilde{Z}_V$
Step 1: Topology Modeling & Filling
for $v_i \in \tilde{Z}_V$ do
    Retrieve related facts $\mathcal{F}_i$ via similarity scoring
    Update $\tilde{z}_i$ with $\mathcal{F}_i$ using $f_{agg}$
end for
Step 2: Graph Topology Reasoning
for $l = 1, \ldots, L$ do
    Message passing layer to update $Z_V$
    Attention distillation over $G$
end for
Step 3: LLM Integration & Training
Flatten $Z_V$ and pack into sequences
Create auxiliary prompts $p_\phi$ with $A_\phi$
Jointly train $M_\theta$ on sequences using $p_\phi$
Update $A_\phi$ using policy gradient
```

Algorithm 2 Knowledge Filtering Module

```
Input: $x$ (Input text)
Output: $\{y, n\}$ (Retrieve), $\{r, ir\}$ (Relevant), $\{c, ic\}$ (Coherent)
Retrieve Module:
Decide whether passage retrieval is needed based on $x$
    if retrieval is needed then
        Set $y$ to Yes
        Set $n$ to No
    else
        Set $y$ to No
        Set $n$ to No retrieval needed
    end if
Relevant Module:
    Filter out irrelevant passages based on $x$ and $p$ (Retrieved passage)
    Set $r$ to Relevant passages
    Set $ir$ to Irrelevant passages
Coherent Module:
    Verify coherence between generated response and input $x$ and $y$
    Set $c$ to Coherent response
    Set $ic$ to Incoherent response
```

Input Tell me more about the painter Van Gogh
Output [IFReT =Yes]
Reference [35]

[IFReL = **Relevant]** Van Gogh was a famous Dutch painter during the post-impressionist period. [IFSuP = **Partially Supported]** He was known for paintings like the Starry Night and Sunflowers.

[IFReT = **Continue]** However, he struggled with mental illness and remained poor and virtually unknown throughout his life. [IFSeP = **Fully Supported]** Most of his best works were produced in the last two years before his death. **[** IFUsE **=4]**

Reference [35] Vincent Willem van Gogh was a Dutch post-impressionist painter who is among the most famous and influential figures in the history of Western art. In a decade, he created about 2,100 artworks, including around 860 oil paintings, most of which date from the last two years of his life. They include landscapes, still lifes, portraits and self-portraits, and are characterised by bold colours and dramatic, impulsive and expressive brushwork that contributed to the foundations of modern art. He was little known during his lifetime, and many of his masterpieces were not appreciated until the 20th century. By the late 1920s, he had become one of the most celebrated artists in history. Nonetheless, mental illness plagued him throughout his life, and after he shot himself in the chest with a revolver at age 37, he succumbed to his injuries two days later.

Correctness of IFReL **and** IFSuP IFSuP is incorrect in claiming full support about his lack of fame during life, when the reference clearly states he only gained appreciation after death. So IFSuP should be partially supported. IFReL is appropriately marked as relevant overall.

C Visualization of Engineering Application

The application of Struct-X in financial report analysis demonstrates its significant potential for industrial-scale data processing and decision support. This case study illustrates how Struct-X's key components - Retrieval, Self-Reg, and Auxiliary Module - synergistically enhance the model's ability to handle complex financial tasks with high accuracy and efficiency for industry application.

In the context of financial analysis, Struct-X showcases its versatility across various critical tasks. The Retrieval component proves invaluable in gathering historical data and industry benchmarks, enabling more comprehensive and contextual analyses. This is particularly evident in tasks such as year-over-year performance evaluation and earnings forecasting, where external data integration is crucial for accurate assessments.

Our Self-Reg mechanism demonstrates its effectiveness in focusing Struct-X's attention on the most relevant financial metrics and accounting policies. This selective attention is vital in an industry where information overload can lead to misinterpretations or oversight of critical details. By filtering out noise and highlighting key information, Self-Reg contributes to more targeted analyses.

The Auxiliary Module emerges as a critical component in guiding complex, multi-step reasoning processes. Its role in ensuring logical consistency and facilitating holistic evaluations is particularly noteworthy in tasks like comprehensive financial health analysis and forecasting. This module effectively mimics the structured thinking process of financial experts, enhancing the model's ability to draw nuanced conclusions from complex data sets.

By combining advanced natural language processing with domain-specific reasoning capabilities, Struct-X represents a significant advancement in AI-driven financial analytics, aligning closely with the industry's growing need for sophisticated, reliable, and efficient analytical tools.

Financial Report Analysis Case Study

Please indicate which Struct-X components (Retrieval, Self-Reg, Auxiliary Module) are crucial for industry application. Here we provide a brief explanation.

Instruction: Analyze Q2 2024 financial report to identify key performance indicators and their year-over-year changes.
Key components: [Retrieval, Self-Reg]
Explanation: Struct-X uses Retrieval to gather historical data for comparison. Self-Reg filters out irrelevant financial metrics, focusing on key indicators for more accurate year-over-year analysis.
Instruction: Evaluate the consistency of revenue recognition methods across different product lines.
Key components: [Self-Reg, Auxiliary Module]
Explanation: Self-Reg helps in identifying relevant accounting policies. The Auxiliary Module guides logical comparisons across product lines, ensuring consistent evaluation of revenue recognition methods.
Instruction: Forecast next quarter's earnings based on current financial data and market trends.
Key components: [Retrieval, Auxiliary Module]
Explanation: Retrieval gathers market trend data. The Auxiliary Module guides the forecasting process, integrating financial data with market trends for a more accurate prediction.
Instruction: Identify potential discrepancies in the cash flow statement and suggest areas for auditor focus.
Key components: [Self-Reg, Auxiliary Module]
Explanation: Self-Reg filters and focuses on relevant cash flow components. The Auxiliary Module guides the logical process of identifying inconsistencies and determining audit priorities.
Instruction: Conduct a comprehensive analysis of the company's financial health, including liquidity, solvency, and profitability.
Key components: [Retrieval, Self-Reg, Auxiliary Module]
Explanation: Retrieval gathers industry benchmarks. Self-Reg focuses on critical financial ratios. The Auxiliary Module guides the multi-step analysis process, ensuring a holistic evaluation of the company's financial health.