Knowledge Graph Prompting for Multi-Document Question Answering

Yu Wang,[1] **Nedim Lipka,**[2] **Ryan A. Rossi,**[2] **Alexa Siu,**[2] **Ruiyi Zhang,**[2] **Tyler Derr**[1]

[1] Vanderbilt University, Nashville, USA
[2] Adobe Research, San Jose, USA
yu.wang.1@vanderbilt.edu, {lipka, ryrossi, asiu, ruizhang}@adobe.com, tyler.derr@vanderbilt.edu

Abstract

The 'pre-train, prompt, predict' paradigm of large language models (LLMs) has achieved remarkable success in open-domain question answering (OD-QA). However, few works explore this paradigm in multi-document question answering (MD-QA), a task demanding a thorough understanding of the logical associations among the contents and structures of documents. To fill this crucial gap, we propose a Knowledge Graph Prompting (KGP) method to formulate the right context in prompting LLMs for MD-QA, which consists of a graph construction module and a graph traversal module. For graph construction, we create a knowledge graph (KG) over multiple documents with nodes symbolizing passages or document structures (e.g., pages/tables), and edges denoting the semantic/lexical similarity between passages or document structural relations. For graph traversal, we design an LLM-based graph traversal agent that navigates across nodes and gathers supporting passages assisting LLMs in MD-QA. The constructed graph serves as the global ruler that regulates the transitional space among passages and reduces retrieval latency. Concurrently, the graph traversal agent acts as a local navigator that gathers pertinent context to progressively approach the question and guarantee retrieval quality. Extensive experiments underscore the efficacy of KGP for MD-QA, signifying the potential of leveraging graphs in enhancing the prompt design and retrieval augmented generation for LLMs. Our code: https://github.com/YuWVandy/KG-LLM-MDQA.

Introduction

Due to the emergence of large language models (LLMs), the 'pre-train, prompt, and predict' paradigm has revolutionized natural language processing (NLP) in real-world applications, such as open-domain question answering, fact-checking, and arithmetic reasoning (Chen et al. 2017; Thorne et al. 2018; Asai et al. 2019; Karpukhin et al. 2020; Aly et al. 2021; Qin et al. 2023; Zou and Caragea 2023; Liu, Dong, and Zhang 2023). However, no significant efforts have investigated this framework in the scenario of multi-documental question answering (MD-QA), which enjoys practical usage in academic research, customer support, and financial/legal inquiries that require deriving insightful analysis from multiple documents (Tessuto 2011; Bolino, Long, and Turnley 2016).

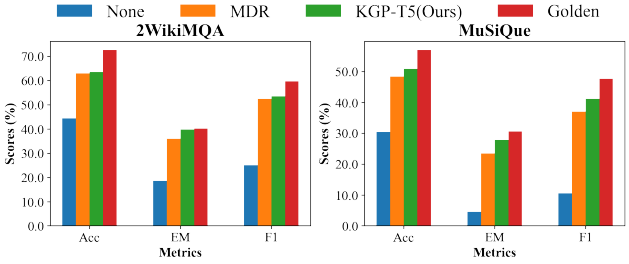


Figure 1: MD-QA performance when prompting ChatGPT with the context retrieved using different strategies.

To investigate the capability of LLMs for MD-QA, we randomly sample multi-document questions from the development set of 2WikiMQA (Ho et al. 2020) and MuSiQue (Trivedi et al. 2022b), and then prompt LLMs in four different strategies for the answer[1]. Successfully answering these questions requires knowledge from multiple Wikipedia documents. As shown in Figure 1, on 2WikiMQA and MuSiQue, directly prompting LLMs without providing any context, i.e., None, performs far worse than when prompting with supporting facts [2] provided as contexts, i.e., the Golden one. This demonstrates the limitation of fulfilling MD-QA using solely the knowledge encoded in LLMs. One common solution to overcome this limitation in conventional OD-QA and single document question-answering (D-QA) (Xu et al. 2020; Mathew, Karatzas, and Jawahar 2021) is to retrieve grounding contexts and derive faithful answers from the contexts, i.e., retrieve-and-read (Zhu et al. 2021; Ju et al. 2022). However, unlike OD-QA and D-QA, the primary challenge of MD-QA roots in its demands for alternatively retrieving and reasoning knowledge across different documents (Pereira et al. 2023; Caciularu et al. 2023). For example, successfully answering questions in Figure 2(a)-(b) requires reasoning over distinct passages from two different documents (i.e., Wikipedia pages). Moreover, each document is a compilation of multi-modality structured data (e.g., pages, sections, paragraphs, tables, and figures) and some questions may specifically ask for the content in cer-

[1] Detailed experimental setting is presented in Section 13.
[2] Supporting facts: passages assumed to contain the answer.

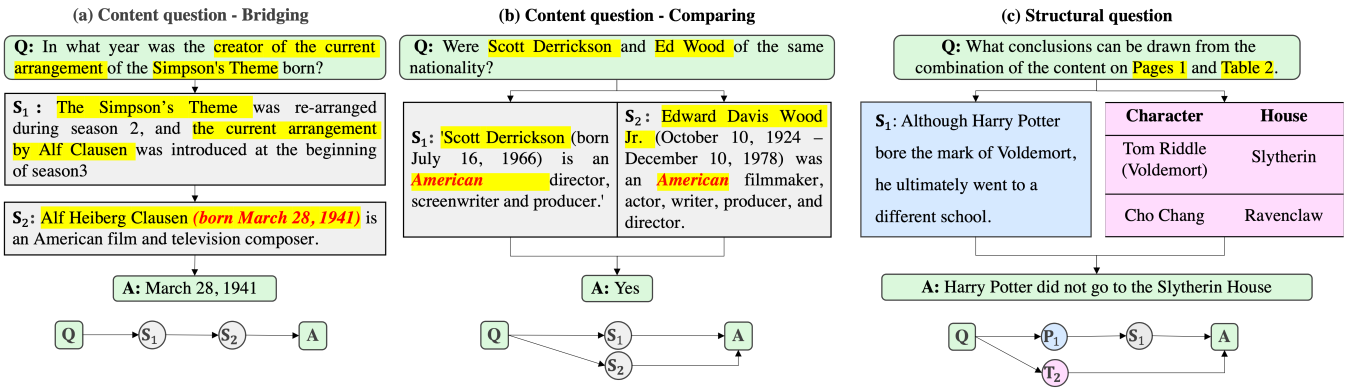


Figure 2: Three popular questions that require reasoning and retrieving over passages/pages/tables from multiple documents. (a) Bridging questions rely on sequential reasoning while (b) Comparing questions rely on parallel reasoning over different passages. (c) Structural questions rely on fetching contents in the corresponding document structures.

tain structures, which necessitates a comprehensive grasp of these complex document structures. For example, the question in Figure 2(c) asks about the difference between Page 1 and Table 2, which is unanswerable if using heuristic methods like BM25 or deep-learning ones like DPR (Karpukhin et al. 2020). Building on top of previous challenges, the advent of LLMs introduces new complexities.

For the challenge of alternatively retrieving and reasoning knowledge across different documents, although previous works train a multi-hop retriever (Xiong et al. 2020; Yavuz et al. 2022) to imitate such process by sequentially fetching the next passage based on the already-retrieved ones, none of them explore the potential of engaging LLMs into this process. More recent works design different prompting strategies such as Chain/Tree/Graph-of-thought (Trivedi et al. 2022a; Wei et al. 2022; Yao et al. 2023; Yao, Li, and Zhao 2023) to guide LLMs approaching answers progressively. However, prompting non-open-sourced LLMs back and forth incurs forbiddable latency as well as unaffordable consumption. In addition, how to integrate different document structures into the prompt design so that LLMs can understand them is still an open-ended question.

Given the above challenges, we propose a knowledge graph prompting (KGP) method for enhancing LLMs in MD-QA. Specifically, we construct a KG over the given documents with nodes symbolizing passages or document structures and edges denoting their lexical/semantic similarity between passages or intra-document structural relations. Then for the first challenge of alternative reasoning and retrieving knowledge across different documents, we design an LLM-based KG traversal agent, which can alternatively generate the next evidence to approach the question, i.e., reasoning, and select the most promising neighbor to visit from the constructed KG based on the generated evidence, i.e., retrieval. Moreover, we apply the instruction fine-tuning strategy to augment the reasoning capability of the LLM-based KG traversal agent and hence refrain from repeatedly prompting non-open-sourced LLMs for evidence generation. For the multi-modality challenge, we add different types of nodes to the KG characterizing different document structures and hence enabling content retrieval within those specific structures. We highlight our contributions as follows:

- **Generally-applicable KG Construction.** We propose three KG construction methods over documents, with passages or document structures as nodes and their lexical/semantical similarity or structural relations as edges. Then we empirically evaluate the quality of the constructed KGs in MD-QA by checking the level of overlap between the neighborhood and the supporting facts for each question (Figure 5). Additionally, we provide a comprehensive summary of both our proposed and existing KG construction methods in Table 5 in Supplementary.
- **Engaging KG for Prompt Formulation.** We design a Knowledge Graph Prompting (KGP) method, which leverages the LLM-based KG traversal agent to retrieve the question-relevant contexts by traversing the constructed KG. Moreover, we fine-tune this agent to adaptively traverse the most promising neighbors for approaching the question based on the visited nodes (retrieved passages).
- **Case Studies Verifying MD-QA Framework.** We compare the performance of MD-QA when using different types of LLM agents in graph traversal (Table 2) on the KGs constructed over different numbers of documents (Figure 7(c)). We conduct case studies on visualizing KGP for MD-QA in Section 8.7 in Supplementary.

Notations

Following (Tian et al. 2023a), let $G = (\mathcal{V}, \mathcal{E})$ be a knowledge graph constructed from a set of documents $\mathcal{D}$, where the node set $\mathcal{V} = \{v_i\}_{i=1}^n$ representing document structures (e.g., passages/pages/tables, etc.) and the edge set $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ representing the connections among different nodes (e.g., semantic/lexical similarity and belonging relations among document structures, etc.). Let $\mathcal{X} = \{\mathcal{X}_i\}_i^n$ be node features and $\mathcal{X}_i$ corresponds to the feature of node v_i, the form of which could be the text for the passage, the markdown for the table and the page number for the page.

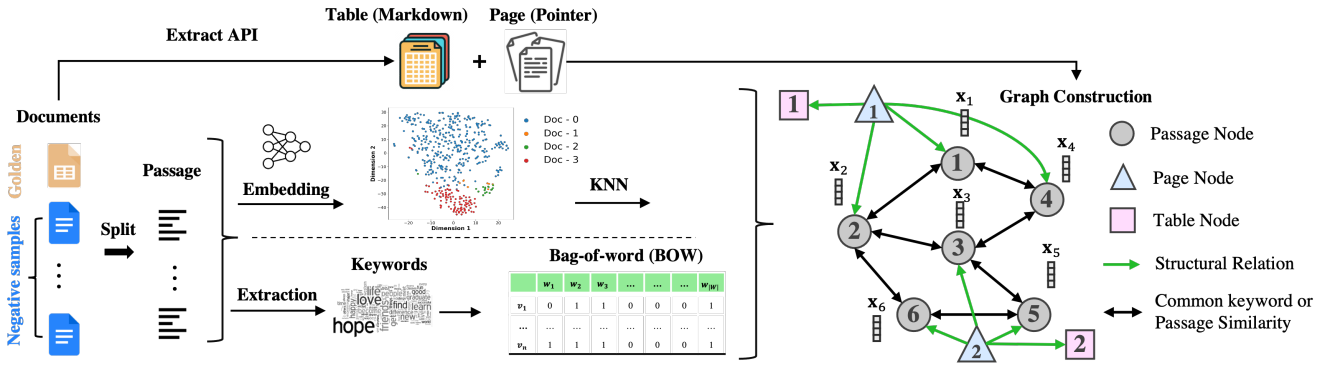


Figure 3: Knowledge Graph Construction. We split each document in the document collection into passages. For each passage, we either directly obtain their embeddings via pre-trained encoders or extract their keywords to build bag-of-word (BOW) features. Then we connect two passages based on their embedding similarity or whether they share common keywords. Additionally, we extract tables/pages via Extract-PDF API and add them as structural nodes to the KG. If pages include passages and tables, we add a directed edge to denote the belonging relations. The table nodes include the markdown formatted content of that table as Figure 8 in Supplementary has empirically shown that LLMs are able to understand tables in this format.

Knowledge Graph Construction

Despite numerous well-established KGs (Hoffart et al. 2013; Tian et al. 2023b), they treat nodes/edges as entities/relations, which necessitates sophisticated relational extraction techniques and thereby limits their applicability in general domains (Huang et al. 2021). Additionally, their primary focus on the Wikipedia domain also restricts their usage for answering non-Wikipedia questions such as ones over legal or financial documents. To remedy this issue, we propose generally applicable KG construction methods.

We first analyze two representative questions in Figure 2(a)-(b) to motivate our KG construction. Answering these two questions necessitates the deduction of logical associations among different passages. These associations are encoded either through 1) lexical similarity: common keywords shared among different passages, e.g., 'Alf Clausen' bridges passage $\mathbf{S}_1$ and passage $\mathbf{S}_2$ in Figure 2(a), or 2) semantic similarity: syntactic elements that convey semantic relations, e.g., 'nationality' and 'American director' in Figure 2(b). This motivates us to construct the graph by modeling passages as nodes and their lexical/semantic similarity as edges. More specifically in Figure 3, we split each document into individual passages, and for each passage $\mathbf{S}_i$, we add a node v_i to the KG with its feature being the text of that passage $\mathcal{X}_i$. Then we add edges by checking the lexical/semantic similarity between pairs of passage nodes.

TF-IDF KG Construction For adding edges according to lexical similarity, we first apply TF-IDF (Ramos et al. 2003) keyword extraction and filtering over each document, which reduces the dimension of bag-of-word (BOW) features, sparsifies the constructed graph and increases the graph traversal efficiency. In addition, we add the document title into the extracted keyword set since some questions focus on title entities. We collect the extracted keywords from all documents to form the keyword space $\mathcal{W}$ and then connect two passages if they share any common keyword in $\mathcal{W}$.

KNN-ST/MDR KG Construction For adding edges according to semantic similarity, we can readily employ pre-existing models such as sentence transformers to generate passage embedding $\mathbf{X}_i$ for each node v_i and subsequently compute pairwise similarity matrix to construct the K-nearest neighbor (KNN) graph. However, these off-the-shelf models, typically trained on tasks not so-related to MD-QA, may not adequately encapsulate necessary logical associations in their embedding similarity demanded by the question. To overcome this problem, we follow the training strategy of MDR (Xiong et al. 2020) and train a sentence encoder by predicting the subsequent supporting facts based on previously supporting facts, thereby endowing the encoder with reasoning capability. Consequently, the embedding similarity and the corresponding constructed KNN graph fundamentally encapsulate the necessary logical associations between different passages.

TAGME Moreover, we employ TAGME (Min et al. 2019) to extract Wikipedia entities from each passage and construct the graph based on whether two passage nodes share common Wikipedia entities.

In addition to passage nodes, we further add structural nodes into the graph by extracting document structures via Extract-PDF [3]. In this paper, we only consider adding pages and tables but the constructed KG can include more different types of document structures. The feature of table nodes is the markdown since LLMs can understand this as demonstrated in Figure 8 in Supplementary. The feature of page nodes is the page number and we add directed edges from it to sentence/table nodes in that page. *Note that we do not aim to propose a one-size-fits-all KG construction method. Instead, we seek to compare the merits and limitations of various methods in Table 5, offering guidance on which KGs are best suited for specific scenarios.*

[3] https://developer.adobe.com/document-services/docs/overview/pdf-extract-api/

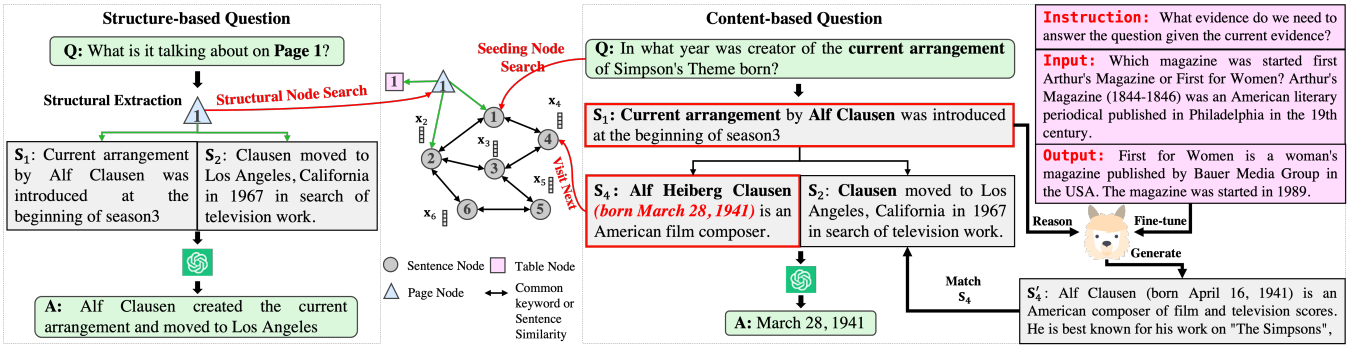


Figure 4: LLM-based KG traversal agent for context retrieval. For questions on document structures (left), we employ LLM to extract structures and retrieve their corresponding contents (the content of pages are passages belonging to that page and the content of tables is the markdown-formatted text). For questions on document content, we concatenate it with the currently retrieved context and prompt the LLM to generate the next evidence to answer the question. By comparing the similarity between the candidate neighboring sentences and the generated passage, we determine the next passage node to traverse. Correspondingly, the candidate neighbors are updated for the next round of traversal.

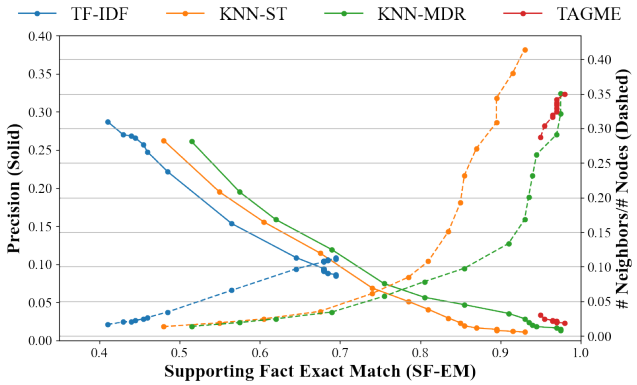


Figure 5: Quality of KGs on HotpotQA. For each KG Construction method, as the average number of neighbors increases (KG becomes denser) in the right y-axis, the SF-EM increases while the precision decreases. KNN-MDR achieves a better trade-off than TF-IDF and KNN-ST. KGs constructed by TAGME are denser than others.

To verify the constructed KGs indeed encode the necessary information for MD-QA, we randomly sample questions from HotpotQA and construct KGs over the set of documents for each of these questions using our proposed methods. We vary the hyperparameters to control the sparsity of the constructed KG and measure how much percentage of the supporting facts are covered by neighbors of the seeding passages initially searched by TF-IDF. More details about the four construction methods and their hyperparameters are included in Section 8.5 in Supplementary. As shown in Figure 5, as the constructed graph becomes denser, the chance that the neighboring node passages hit the supporting facts increases (i.e., SF-EM increases) although the redundant information also increases (i.e., the precision decreases). Given the common keywords shared between one passage to all other passages are typically far less than the total number of passages across all documents, the density of the constructed graph by TF-IDF would be upper-bounded, causing lower SF-EM (evidenced by SF-EM below 0.7 in Figure 5 for TF-IDF curve). For TAGME, we empirically find it identifies a larger quantity of entities mentioned in a single passage, which leads to a denser graph and causes the starting SF-EM of TAGME to be already around 0.95. In addition, since KNN-MDR is pre-trained by predicting the next supporting facts (Xiong et al. 2020) on HotpotQA, it achieves better trade-off than KNN-ST where the embeddings are directly obtained from the sentence transformer without dataset-specific pre-training.

To summarize, although high SF-EM indicates that the supporting facts for most questions are fully covered by the neighbors of seeding passages, low precision signifies that most of these neighboring passages are irrelevant to the question. Therefore, if we blindly perform graph traversal without any question-tailored adaptation, our retrieved contexts would include redundant passages and compromise the capability of LLMs in MD-QA. To remedy this issue, we next introduce an LLM-based KG traversal agent to adaptively visit neighboring passages that are most conducive to answering the given question.

LLM-based KG Traversal Agent

One way to enable adaptive knowledge graph traversal is to rank the candidate nodes, i.e., the already-visited nodes' neighbors, to determine which nodes to visit next. The simplest way is to apply heuristic-based fuzzy matching or embedding-based similarity ranking, which cannot capture the intrinsic logical relations between the already traversed paths and the nodes to visit next. Instead, we introduce an LLM-based KG traversal agent, which is a fine-tuned LLM to guide the KG traversal toward the next most promising passages for answering the question based on the information collected from the currently visited nodes.

Given a question q asking about the document content, the LLM-based graph traversal agent reasons over previously

visited nodes/retrieved passages $\{s_k\}_{k=0}^{j}$ and then generates the next passage s_{j+1} as follows:

$$s_{j+1} = \arg\max_{v \in \mathcal{N}_j} \phi(g(\mathcal{X}_v), f(||_{k=0}^{j} \mathcal{X}_k)), \quad (1)$$

where $||_{k=0}^{j} \mathcal{X}_k$ concatenates the textual information of previously retrieved passages/visited nodes. For the choice of f, one way is to employ encoder-only models like Roberta-base (Asai et al. 2019; Xiong et al. 2020; Yavuz et al. 2022) and correspondingly g would be another encoder model with $\phi(\cdot)$ being the inner product measuring the embedding similarity. Another way is to employ encoder-decoder models such as T5 (Brown et al. 2020; Touvron et al. 2023) and correspondingly g would be an identity function with $\phi(\cdot)$ measuring the textual similarity. To mitigate the hallucination issue and enhance the reasoning capability (Wei et al. 2022; Ji et al. 2023) of the LLM traversal agent, we further instruction fine-tune f (Chung et al. 2022) by predicting the next supporting facts based on previous supporting facts, thereby integrating commonsense knowledge encoded originally in their pre-trained parameters with the enhanced reasoning capability inherited from the instruction fine-tuning. After visiting the top-scoring nodes selected from the candidate neighbor queue by Eq (1), the candidate neighbor queue is updated by adding neighbors of these newly visited nodes. We iteratively apply this process until hit the preset budget. Next, we illustrate the above process with an example in Figure 4 and present the algorithm thereafter.

Figure 4 presents the content-based question asking 'In what year was the creator of the current arrangement of Simpson's Theme born?'. We use TF-IDF search to initialize the seeding passage Node 1, which reads: 'Alf Heiberg Clausen (born March 28, 1941) is an American film composer'. Subsequently, we prefix the currently retrieved-context (Node 1) with the question and prompt the LLM to generate the next evidence required to approach the question one step closer. Because we augment the reasoning capability of the LLM by instruction fine-tuning, it is expected to recognize the logical association between the question and the currently retrieved context. Consequently, it can predict the subsequent passage that *maintains logical coherence, albeit may contain factual mistakes*, i.e., 'Alf Clausen (born April 16, 1941) is an American composer of film and television scores.' To rectify this potential factual mistake, we select nodes from the candidate neighbors that match the most with the LLM-generated passage, in this case, Node 4 'Alf Heiberg Clausen (born March 28, 1941) is an American film composer'. Since this passage sources directly from documents, it inherently ensures the validity of the information. Then we prompt LLMs along with the retrieved context Node 1 and 4 for the answer.

Additionally, for questions asking about document structures, we extract the document structure names and locate their corresponding structural nodes in the KG. For the table node, we retrieve its markdown formatted content while for the page node, we traverse its one-hop neighbor and obtain passages belonging to that page.

Here we present the algorithm for our proposed KGP method for MD-QA. Given a question, we first apply LLM to classify whether the question is asking about the document structure or content. If the question focuses on the document structure, we extract the structural keywords such as Page or Table, and retrieve the content in the corresponding structural nodes in KG. If the question focuses on the document content, we follow the step according to Algorithm 1. Specifically, we first initialize seeding passages $\mathcal{V}^s$ and the reasoning path queue $\mathcal{P}$ by TF-IDF search. Then for each seeding passage $v_i \in \mathcal{V}^s$, we add its neighboring passage nodes $\mathcal{N}_i$ into the candidate neighbor queue $\mathcal{C}$ (lines 1-4). After that, we iteratively dequeue the earliest enqueued reasoning path/candidate neighborhood $\mathcal{P}_i/\mathcal{C}_i$ from $\mathcal{P}/\mathcal{C}$ and employ the fine-tuned LLM-based graph traversal agent to rank the dequeued neighbors in $\mathcal{C}_i$ by Eq. (1) (lines 5-7). Last, we select top-k passage nodes $\mathcal{V}_i'$ from $\mathcal{C}_i$ to visit next based on their rank and correspondingly update the candidate neighbor queue and reasoning path queue (lines 8-13). The above process terminates when either the candidate neighbor queue becomes empty or the prefixed budget K for the retrieved passages is met. The time and space complexity are thoroughly analyzed in Section 8.3 in Supplementary.

Algorithm 1: LLM-based KG Traversal Algorithm to Retrieve Relevant Context for Content-based Question.

Input: A question q over a set of documents $\mathcal{D}$, the constructed KG $G = \{\mathcal{V}, \mathcal{E}, \mathcal{X}\}$ over $\mathcal{D}$, the fine-tuned LLM-guided graph traversal f_{GT}, the preset context budget K, the TF-IDF search function g.

1 Initialize seed passages $\mathcal{V}^s = g(\mathcal{V}, \mathcal{X}, q)$
2 Initialize the retrieved passage queue $\mathcal{P} = [\{v_i\} | v_i \in \mathcal{V}^s]$
3 Initialize the candidate neighbor queue $\mathcal{C} = [\mathcal{N}_i | v_i \in \mathcal{V}^s]$
4 Initialize the retrieved passage counter $k = \sum_{\mathcal{P}_i \in \mathcal{P}} |\mathcal{P}_i|$
5 **while** *queue $\mathcal{P}$ and queue $\mathcal{C}$ are not empty* **do**
6 $\quad \mathcal{P}_i \leftarrow \mathcal{P}$.dequeue(), $\mathcal{C}_i \leftarrow \mathcal{C}$.dequeue()
7 $\quad \mathcal{V}_i' =$ Graph Traversal$(\{q\} \cup \mathcal{P}_i, \mathcal{C}_i, k)$ by Eq (1)
8 $\quad$ **for** $v \in \mathcal{V}_i'$ **do**
9 $\quad\quad \mathcal{P}$.enqueue$(\mathcal{P}_i \cup \{v\})$, $\mathcal{C}$.enqueue$(\mathcal{N}_v)$
10 $\quad\quad k \leftarrow k + 1$
11 $\quad\quad$ **if** $k > K$ **then**
12 $\quad\quad\quad$ **Terminate**
13 **return** *Retrieved Passage Queue $\mathcal{P}$*

Experiment

In this section, we conduct experiments to verify the proposed knowledge graph prompting method (KGP) for MD-QA. In particular, we answer the following questions:

- **Q1 - Section 13**: How well does KGP perform MD-QA compared with existing baselines?
- **Q2 - Section 13-13**: How do the quality of the constructed KG and the LLM-based graph traversal agent impact the MD-QA performance?

Due to the space limitation, we comprehensively introduce our experimental setting, including dataset collection, baselines, and evaluation criteria, in Supplementary 8.1-8.2.

Method	HotpotQA			IIRC			2WikiMQA			MuSiQue			PDF-T	Rank	
	Acc	EM	F1	Acc	EM	F1	Acc	EM	F1	Acc	EM	F1	Struct-EM	w PDF-T	w/o PDF-T
None	41.80	19.00	30.50	19.50	8.60	13.17	44.40	18.60	25.07	30.40	4.60	10.58	0.00	8.53	9.00
KNN	71.57	40.73	57.97	43.82	25.15	37.24	52.40	31.20	42.13	44.70	18.86	30.04	–	7.00	7.33
TF-IDF	**76.64**	45.97	64.64	47.47	27.22	40.80	58.40	34.60	44.50	44.40	21.59	32.50	–	4.85	5.00
BM25	71.95	41.46	59.73	41.93	23.48	35.55	55.80	30.80	40.55	44.47	21.11	31.15	–	6.92	7.25
DPR	73.43	43.61	62.11	48.11	26.89	41.85	62.40	35.60	51.10	44.27	20.32	31.64	–	5.31	5.50
MDR	75.30	45.55	65.16	**50.84**	27.52	**43.47**	63.00	36.00	52.44	48.39	23.49	37.03	–	3.07	3.08
IRCoT	74.36	45.29	64.12	49.78	**27.73**	41.65	61.81	37.75	50.17	45.14	22.46	34.21	–	4.00	4.08
KGP-T5	76.53	**46.51**	**66.77**	48.28	26.94	41.54	**63.50**	**39.80**	**53.50**	**50.92**	**27.90**	**41.19**	**67.00**	**2.69**	**2.75**
Golden	82.19	50.20	71.06	62.68	35.64	54.76	72.60	40.20	59.69	57.00	30.60	47.75	100.00	1.00	1.00

Table 1: MD-QA Performance (%) of different baselines. The best and runner-up are in bold and underlined. None: no passages but only the question is provided. Golden: supporting facts are provided along with the question. PDF-T stands for PDFTriage.

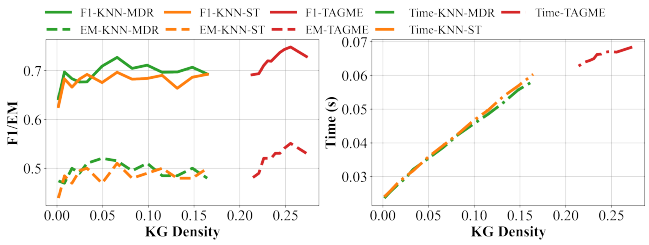


Figure 6: The performance/latency increases as the KG density increases. The results are averaged across 100 randomly sampled questions on HotpotQA.

Traversal Agent	HotpotQA			2WikiMQA			MuSiQue		
	Acc	EM	F1	Acc	EM	F1	Acc	EM	F1
TF-IDF	73.5	43.8	63.1	58.1	35.1	46.0	44.7	21.9	32.9
MDR	75.7	46.1	65.8	60.9	37.2	51.3	**51.2**	27.8	41.1
ChatGPT	**77.8**	46.0	66.6	61.6	36.2	49.4	50.6	26.9	38.7
LLaMA	75.7	46.2	66.3	62.5	37.6	52.5	50.8	26.7	40.0
T5	76.5	**46.5**	**66.8**	**63.5**	**39.8**	**53.5**	50.9	**27.9**	**41.2**

Table 2: Comparing LLM-based KG Traversal Agents.

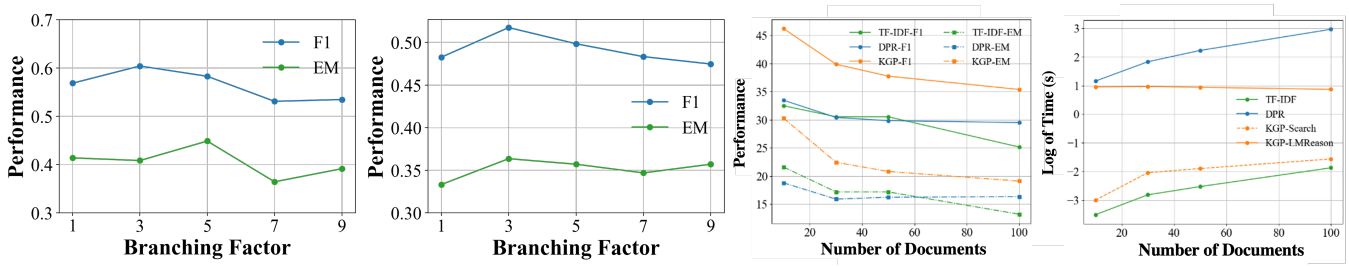
Performance Comparison on MD-QA

We compare the MD-QA performance of the proposed KGP-T5 and other baselines in Table 1. First, baselines 'None/Golden' perform the worst/best because they provide no/golden context. All other baselines achieve the performance in-between because the retrieved context only covers the partial of the golden supporting facts. Our KGP-T5 ranks the best except for the Golden. The 2nd-performing baseline MDR fine-tunes a RoBERTa-base encoder by predicting the next supporting fact based on the question and the already retrieved contexts (Xiong et al. 2020). Their pretext task equips the model with the reasoning capability of the knowledge across different passages and hence increases the quality of the retrieved contexts. The other deep-learning-based retriever DPR achieves much worse performance than MDR since it only fine-tunes the encoder by maximizing the similarity between the query and its supporting facts regardless of their sequential order, demonstrating the importance of understanding the logical order of different knowledge when solving MD-QA (Xiong et al. 2020). When comparing performance across datasets, we find that all baselines perform better on HotpotQA than IIRC. This is because questions in HotpotQA are generally simpler than in IIRC. Existing works (Jiang and Bansal 2019) have shown some questions in HotpotQA can be easily answered following shortcuts while questions in IIRC sometimes necessitate arithmetic skills, e.g., 'How many years did the event last when Wingfield lost his fortune?', which poses unique difficulty due to LLMs' inferior arithmetic capability (Yuan et al. 2023).

Moreover, our proposed method achieves 67% Struct-EM on PDFTriage, whereas no existing baselines are designed to handle these structural questions, e.g., 'What is the difference between Page 1 and Page 2' or 'In Table 3, which station has the highest average flow rate?'.

Impact of the Constructed Graph

We construct KGs with varying densities by varying the hyperparameters of TF-IDF/KNN-ST/KNN-MDR/TAGME, and studying its impact on the performance and the neighbor matching time of MD-QA using KGP-T5. Since the LLM-based graph traversal agent selects the next node to visit from neighbors of already visited nodes, the chance that it hits the supporting facts increases as neighbors increase. In contrast, the neighbor matching efficiency decreases as the candidate pool, i.e., $\mathcal{N}_j$ in Eq (1), increases. As evidenced in Figure 6, we observe a similar trend, i.e., as KG density increases, the F1/EM increases and stays stable while the latency for selecting the most promising neighbors to visit next also increases. KNN-MDR achieves better performance than KNN-ST when the density of the two constructed KGs is the same. This is because the encoder in KNN-ST is pre-trained on wide-spectrum datasets while the encoder in MDR is specifically pre-trained on the HotpotQA by the pretext task of predicting the next supporting facts. Therefore, the embedding similarity and the corresponding neighbor relations better reflect the logical associations among different passages, which aligns with the better constructed KG by KNN-MDR than KG by KNN-ST in Figure 5. Compared with KNN-MDR/ST, TAGME delivers superior performance at the cost of increasing latency since the generated KG by TAGME is denser than KGs by KNN-ST/MDR.



(a) 2WikiMQA (b) MuSiQue (c) Performance and Efficiency on MuSiQue

Figure 7: (a)-(b): Performance first increases and then decreases as the branching factor increases. The results are averaged across 100 sampled questions on 2WikiMQA and MuSiQue. (c): Performance/Efficiency increases/decreases as the number of documents increases on MuSiQue. KGP-T5 achieves higher performance/efficiency than DPR.

Impact of Graph Traversal Agent

Here we study the influence of using different LLM agents to traverse over TAGME-constructed KG on MD-QA. Specifically, we compare agents that select the next neighbor to visit randomly or intelligently via guidance from ChatGPT, LLaMA, T5, and MDR in Table 2. Because the random agent only blindly traverses the KG without any guidance from LLM, it unavoidably collects irrelevant passages and hence achieves the worst performance than others under LLMs' guidance. This aligns with our previous observation on the low precision in Figure 5 and further demonstrates the necessity of using LLMs to guide the graph traversal. Interestingly, we find that KGP-T5 performs better than LLaMA even though the parameters of LLaMA-7B are more than the ones with T5-0.7B. We hypothesize this is because LLaMA-7B requires more data to fine-tune than T5-0.7B.

Sensitivity Analysis

Here we perform the sensitivity analysis of the branching factor (the number of nodes selected from candidate neighbors to visit next). In Figure 7(a)-(b), the performance first increases as the branching factor increases because more passage nodes selected from the candidate neighbors lead to more reasoning paths to reach the final answer. However, as we fix the context budget to ensure fair comparison (i.e., the total number of passages we are allowed to retrieve for each question is the same across all baselines), the performance declines as the branching factor increases because the number of initial seeding nodes diminishes, leading to reduced coverage of the KG. Furthermore, we compare the efficiency of KGP when the constructed KG includes different numbers of documents in Figure 7(c). KGP consistently achieves higher performance than other baselines and higher efficiency than embedding-based DPR. TF-IDF is slightly faster than KGP because it is a purely heuristic-based method.

Related Work

Question answering Question Answering (QA) aims to provide answers to users' questions in natural language (Zhu et al. 2021; Pandya and Bhatt 2021), and most QA systems are composed of information retrieval (IR) and answer extraction (AE) (Mao et al. 2021; Ju et al. 2022; Liu and Qin 2022). In IR, the system searches for query-relevant factual passages using heuristic methods (BM25) (Robertson, Zaragoza et al. 2009) or neural-ranking ones (DPR) (Karpukhin et al. 2020). In AE, the final answer is extracted usually as a textual span from related passages. Although this framework has been broadly applied in O-QA (Mao et al. 2021) and D-QA (Xu et al. 2020; Mathew, Karatzas, and Jawahar 2021), no previous work focus on MD-QA, which demands alternatively reasoning and retrieving knowledge from multiple documents. To tackle this issue, we construct KGs to encode logical associations among different passages across documents and design an LLM-based graph traversal agent to alternatively generate the reason and visit the most matching passage node.

Pre-train, Prompt, and Predict with LLMs With the emergence of LLMs, the paradigm of 'pre-train, prompt, predict' has gained magnificent popularity in handling a wide spectrum of tasks (Gururangan et al. 2020; Liu et al. 2023; Yu et al. 2023). This approach begins with pre-training LLMs by pretext tasks to encode world knowledge into tremendous parameters (Wu et al. 2023) followed by a prompting function to extract pertinent knowledge for downstream tasks (Yang et al. 2023). Recent advancements explore different prompting strategies to enhance LLMs' reasoning capabilities (Wei et al. 2022; Jin et al. 2023). In contrast to that, our work offers a novel perspective by transforming the prompt formulation into the KG traversal.

Conclusion

Answering multi-document questions demands knowledge reasoning and retrieving from different documents across various modalities, presenting challenges for applying the paradigm of 'pre-train, prompt and predict' with LLMs. Recognizing the logical associations among passages and structural relations within documents, we propose a Knowledge Graph Prompting method (KGP) for aiding LLMs in MD-QA. The KGP constructs KGs from documents with nodes as sentences or document structures, and edges as their lexical/semantic similarity/structural relations. As constructed KGs may contain irrelevant neighbors, we further design an LLM-based graph traversal agent that selectively visits the most promising node in approaching the question.

Ethics Statement

Due to page limitation, the supplementary material and reproducing details are publically available at https://github.com/YuWVandy/KG-LLM-MDQA/blob/main/AAAI24_KGPrompting_Supplementary.pdf.

Acknowledgements

This research is supported by Adobe Research and National Science Foundation (NSF) under grant number IIS2239881.

References

Aly, R.; Guo, Z.; Schlichtkrull, M.; Thorne, J.; Vlachos, A.; Christodoulopoulos, C.; Cocarascu, O.; and Mittal, A. 2021. FEVEROUS: Fact extraction and VERification over unstructured and structured information. *arXiv preprint arXiv:2106.05707*.

Asai, A.; Hashimoto, K.; Hajishirzi, H.; Socher, R.; and Xiong, C. 2019. Learning to retrieve reasoning paths over wikipedia graph for question answering. *arXiv preprint arXiv:1911.10470*.

Bolino, M.; Long, D.; and Turnley, W. 2016. Impression management in organizations: Critical questions, answers, and areas for future research. *Annual Review of Organizational Psychology and Organizational Behavior*, 3: 377–406.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Caciularu, A.; Peters, M. E.; Goldberger, J.; Dagan, I.; and Cohan, A. 2023. Peek Across: Improving Multi-Document Modeling via Cross-Document Question-Answering. *arXiv preprint arXiv:2305.15387*.

Chen, D.; Fisch, A.; Weston, J.; and Bordes, A. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Chung, H. W.; Hou, L.; Longpre, S.; Zoph, B.; Tay, Y.; Fedus, W.; Li, E.; Wang, X.; Dehghani, M.; Brahma, S.; et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Gururangan, S.; Marasović, A.; Swayamdipta, S.; Lo, K.; Beltagy, I.; Downey, D.; and Smith, N. A. 2020. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*.

Ho, X.; Nguyen, A.-K. D.; Sugawara, S.; and Aizawa, A. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*.

Hoffart, J.; Suchanek, F. M.; Berberich, K.; and Weikum, G. 2013. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artificial intelligence*, 194: 28–61.

Huang, X.; Zhang, J.; Xu, Z.; Ou, L.; and Tong, J. 2021. A knowledge graph based question answering method for medical domain. *PeerJ Computer Science*, 7: e667.

Ji, Z.; Lee, N.; Frieske, R.; Yu, T.; Su, D.; Xu, Y.; Ishii, E.; Bang, Y. J.; Madotto, A.; and Fung, P. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12): 1–38.

Jiang, Y.; and Bansal, M. 2019. Avoiding reasoning shortcuts: Adversarial evaluation, training, and model development for multi-hop QA. *arXiv preprint arXiv:1906.07132*.

Jin, B.; Liu, G.; Han, C.; Jiang, M.; Ji, H.; and Han, J. 2023. Large Language Models on Graphs: A Comprehensive Survey. *arXiv preprint arXiv:2312.02783*.

Ju, M.; Yu, W.; Zhao, T.; Zhang, C.; and Ye, Y. 2022. Grape: Knowledge graph enhanced passage reader for open-domain question answering. *arXiv preprint arXiv:2210.02933*.

Karpukhin, V.; Oğuz, B.; Min, S.; Lewis, P.; Wu, L.; Edunov, S.; Chen, D.; and Yih, W.-t. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Liu, H.; and Qin, Y. 2022. Heterogeneous graph prompt for community question answering. *Concurrency and Computation: Practice and Experience*, e7156.

Liu, P.; Yuan, W.; Fu, J.; Jiang, Z.; Hayashi, H.; and Neubig, G. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9): 1–35.

Liu, X.; Dong, Z.; and Zhang, P. 2023. Tackling Data Bias in MUSIC-AVQA: Crafting a Balanced Dataset for Unbiased Question-Answering. *arXiv preprint arXiv:2310.06238*.

Mao, Y.; He, P.; Liu, X.; Shen, Y.; Gao, J.; Han, J.; and Chen, W. 2021. RIDER: Reader-Guided Passage Reranking for Open-Domain Question Answering. *arXiv preprint arXiv:2101.00294*.

Mathew, M.; Karatzas, D.; and Jawahar, C. 2021. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2200–2209.

Min, S.; Chen, D.; Zettlemoyer, L.; and Hajishirzi, H. 2019. Knowledge guided text retrieval and reading for open domain question answering. *arXiv preprint arXiv:1911.03868*.

Pandya, H. A.; and Bhatt, B. S. 2021. Question answering survey: Directions, challenges, datasets, evaluation matrices. *arXiv preprint arXiv:2112.03572*.

Pereira, J.; Fidalgo, R.; Lotufo, R.; and Nogueira, R. 2023. Visconde: Multi-document QA with GPT-3 and Neural Reranking. In *European Conference on Information Retrieval*, 534–543. Springer.

Qin, C.; Zhang, A.; Zhang, Z.; Chen, J.; Yasunaga, M.; and Yang, D. 2023. Is ChatGPT a general-purpose natural language processing task solver? *arXiv preprint arXiv:2302.06476*.

Ramos, J.; et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, 29–48. Citeseer.

Robertson, S.; Zaragoza, H.; et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4): 333–389.

Tessuto, G. 2011. Legal problem question answer genre across jurisdictions and cultures. *English for Specific Purposes*, 30(4): 298–309.

Thorne, J.; Vlachos, A.; Christodoulopoulos, C.; and Mittal, A. 2018. FEVER: a large-scale dataset for fact extraction and VERification. *arXiv preprint arXiv:1803.05355*.

Tian, Y.; Dong, K.; Zhang, C.; Zhang, C.; and Chawla, N. V. 2023a. Heterogeneous graph masked autoencoders. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 9997–10005.

Tian, Y.; Song, H.; Wang, Z.; Wang, H.; Hu, Z.; Wang, F.; Chawla, N. V.; and Xu, P. 2023b. Graph neural prompting with large language models. *arXiv preprint arXiv:2309.15427*.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Trivedi, H.; Balasubramanian, N.; Khot, T.; and Sabharwal, A. 2022a. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*.

Trivedi, H.; Balasubramanian, N.; Khot, T.; and Sabharwal, A. 2022b. MuSiQue: Multihop Questions via Single-hop Question Composition. *Transactions of the Association for Computational Linguistics*, 10: 539–554.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35: 24824–24837.

Wu, X.; Zhou, K.; Sun, M.; Wang, X.; and Liu, N. 2023. A survey of graph prompting methods: techniques, applications, and challenges. *arXiv preprint arXiv:2303.07275*.

Xiong, W.; Li, X. L.; Iyer, S.; Du, J.; Lewis, P.; Wang, W. Y.; Mehdad, Y.; Yih, W.-t.; Riedel, S.; Kiela, D.; et al. 2020. Answering complex open-domain questions with multi-hop dense retrieval. *arXiv preprint arXiv:2009.12756*.

Xu, Y.; Xu, Y.; Lv, T.; Cui, L.; Wei, F.; Wang, G.; Lu, Y.; Florencio, D.; Zhang, C.; Che, W.; et al. 2020. Layoutlmv2: Multi-modal pre-training for visually-rich document understanding. *arXiv preprint arXiv:2012.14740*.

Yang, J.; Jin, H.; Tang, R.; Han, X.; Feng, Q.; Jiang, H.; Yin, B.; and Hu, X. 2023. Harnessing the power of llms in practice: A survey on chatgpt and beyond. *arXiv preprint arXiv:2304.13712*.

Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T. L.; Cao, Y.; and Narasimhan, K. 2023. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*.

Yao, Y.; Li, Z.; and Zhao, H. 2023. Beyond Chain-of-Thought, Effective Graph-of-Thought Reasoning in Large Language Models. *arXiv preprint arXiv:2305.16582*.

Yavuz, S.; Hashimoto, K.; Zhou, Y.; Keskar, N. S.; and Xiong, C. 2022. Modeling multi-hop question answering as single sequence prediction. *arXiv preprint arXiv:2205.09226*.

Yu, Y.; Zhuang, Y.; Zhang, J.; Meng, Y.; Ratner, A.; Krishna, R.; Shen, J.; and Zhang, C. 2023. Large Language Model as Attributed Training Data Generator: A Tale of Diversity and Bias. *arXiv preprint arXiv:2306.15895*.

Yuan, Z.; Yuan, H.; Tan, C.; Wang, W.; and Huang, S. 2023. How well do Large Language Models perform in Arithmetic tasks? *arXiv preprint arXiv:2304.02015*.

Zhu, F.; Lei, W.; Wang, C.; Zheng, J.; Poria, S.; and Chua, T.-S. 2021. Retrieving and reading: A comprehensive survey on open-domain question answering. *arXiv preprint arXiv:2101.00774*.

Zou, H.; and Caragea, C. 2023. JointMatch: A Unified Approach for Diverse and Collaborative Pseudo-Labeling to Semi-Supervised Text Classification. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 7290–7301.