Retrieval-Augmented Generation with Knowledge Graphs for Customer Service Question Answering

Zhentao Xu
zhexu@linkedin.com
LinkedIn Corporation
Sunnyvale, CA, USA

Mark Jerome Cruz
marcruz@linkedin.com
LinkedIn Corporation
Sunnyvale, CA, USA

Matthew Guevara
mguevara@linkedin.com
LinkedIn Corporation
Sunnyvale, CA, USA

Tie Wang
tiewang@linkedin.com
LinkedIn Corporation
Sunnyvale, CA, USA

Manasi Deshpande
madeshpande@linkedin.com
LinkedIn Corporation
Sunnyvale, CA, USA

Xiaofeng Wang
xiaofwang@linkedin.com
LinkedIn Corporation
Sunnyvale, CA, USA

Zheng Li
zeli@linkedin.com
LinkedIn Corporation
Sunnyvale, CA, USA

ABSTRACT

In customer service technical support, swiftly and accurately retrieving relevant past issues is critical for efficiently resolving customer inquiries. The conventional retrieval methods in retrieval-augmented generation (RAG) for large language models (LLMs) treat a large corpus of past issue tracking tickets as plain text, ignoring the crucial intra-issue structure and inter-issue relations, which limits performance. We introduce a novel customer service question-answering method that amalgamates RAG with a knowledge graph (KG). Our method constructs a KG from historical issues for use in retrieval, retaining the intra-issue structure and inter-issue relations. During the question-answering phase, our method parses consumer queries and retrieves related sub-graphs from the KG to generate answers. This integration of a KG not only improves retrieval accuracy by preserving customer service structure information but also enhances answering quality by mitigating the effects of text segmentation. Empirical assessments on our benchmark datasets, utilizing key retrieval (MRR, Recall@K, NDCG@K) and text generation (BLEU, ROUGE, METEOR) metrics, reveal that our method outperforms the baseline by 77.6% in MRR and by 0.32 in BLEU. Our method has been deployed within LinkedIn's customer service team for approximately six months and has reduced the median per-issue resolution time by 28.6%.

CCS CONCEPTS

• **Computing methodologies** → **Information extraction**; *Natural language generation.*

KEYWORDS

Large Language Model, Knowledge Graph, Question Answering, Retrieval-Augmented Generation

1 INTRODUCTION

Effective technical support in customer service underpins product success, directly influencing customer satisfaction and loyalty. Given the frequent similarity of customer inquiries to previously resolved issues, the rapid and accurate retrieval of relevant past instances is crucial for the efficient resolution of such inquiries. Recent advancements in embedding-based retrieval (EBR), large language models (LLMs), and retrieval-augmented generation (RAG) [8] have significantly enhanced retrieval performance and question-answering capabilities for the technical support of customer service. This process typically unfolds in two stages: first, historical issue tickets are treated as plain text, segmented into smaller chunks to accommodate the context length constraints of embedding models; each chunk is then converted into an embedding vector for retrieval. Second, during the question-answering phase, the system retrieves the most relevant chunks and feeds them as contexts for LLMs to generate answers in response to queries. Despite its straightforward approach, this method encounters several limitations:

- **Limitation 1 - Compromised Retrieval Accuracy from Ignoring Structures:** Issue tracking documents such as Jira [2] possess inherent structure and are interconnected, with references such as "issue A is related to/copied from/caused

by issue B." The conventional approach of compressing documents into text chunks leads to the loss of vital information. Our approach parses issue tickets into trees and further connects individual issue tickets to form an interconnected graph, which maintains this intrinsic relationship among entities, achieving high retrieval performance.

- **Limitation 2 - Reduced Answer Quality from Segmentation:** Segmenting extensive issue tickets into fixed-length segments to accommodate the context length constraints of embedding models can result in the disconnection of related content, leading to incomplete answers. For example, an issue ticket describing an issue at its beginning and its solution at the end may be split during the text segmentation process, resulting in the omission of critical parts of the solution. Our graph-based parsing method overcomes this by preserving the logical coherence of ticket sections, ensuring the delivery of complete and high-quality responses.

2 RELATED WORK

Question answering (QA) with knowledge graphs (KGs) can be broadly classified into retrieval-based, template-based, and semantic parsing-based methods. Retrieval-based approaches utilize relation extraction [19] or distributed representations [5] to derive answers from KGs, but they face difficulties with questions involving multiple entities. Template-based strategies depend on manually-created templates for encoding complex queries, yet are limited by the scope of available templates [16]. Semantic parsing methods map text to logical forms containing predicates from KGs [4] [14] [21].

Recent advancements in large language models (LLMs) integration with Knowledge Graphs (KGs) have demonstrated notable progress. Jin et al. [7] provide a comprehensive review of this integration, categorizing the roles of LLMs as Predictors, Encoders, and Aligners. For graph-based reasoning, Think-on-Graph [15] and Reasoning-on-Graph [10] enhance LLMs' reasoning abilities by integrating KGs. Yang et al. [20] propose augmenting LLMs' factual reasoning across various training phases using KGs. For LLM-based question answering, Wen et al.'s Mindmap [18] and Qi et al. [13] employ KGs to boost LLM inference capabilities in specialized domains such as medicine and food. These contributions underscore the increasing efficacy of LLM and KG combinations in enhancing information retrieval and reasoning tasks.

3 METHODS

We introduce an LLM-based customer service question answering system that seamlessly integrates retrieval-augmented generation (RAG) with a knowledge graph (KG). Our system (Figure 1) comprises two phases: First, during the KG construction phase, our system constructs a comprehensive knowledge graph from historical customer service issue tickets. It integrates a tree-structured representation of each issue and interlinks them based on relational context. It also generates embedding for each node to facilitate later semantic searching. Second, during the question-answering phase, our method parses consumer queries to identify named entities and intents. It then navigates within the KG to identify related sub-graphs for generating answers.

3.1 Knowledge Graph Construction

3.1.1 Graph Structure Definition. In defining the knowledge graph structure for historical issue representation, we employ a dual-level architecture that segregates intra-issue and inter-issue relations, as illustrated in Figure 1. The **Intra-issue Tree** $\mathcal{T}_i(\mathcal{N}, \mathcal{E}, \mathcal{R})$ models each ticket t_i as a tree, where each node $n \in \mathcal{N}$, identified by a unique combination (i, s), corresponds to a distinct section s of ticket t_i, and each edge $e \in \mathcal{E}$ and $r \in \mathcal{R}$ signifies the hierarchical connection and type of relations between these sections. The **Inter-issue Graph** $\mathcal{G}(\mathcal{T}, \mathcal{E}, \mathcal{R})$ represents the network of connections across different tickets, incorporating both explicit links $\mathcal{E}_{\text{exp}}$, defined in issue tracking tickets, and implicit connections $\mathcal{E}_{\text{imp}}$, derived from semantic similarity between tickets. For implicit connections, we leverage cosine similarity between the embedding vectors of ticket titles, a method adaptable to specific use cases.

For instance, Figure 1 portrays ticket ENT-22970 as a tree structure with nodes representing sections such as Summary, Description, and Priority. It exhibits a direct clone linkage to PORT-133061, indicating an explicit clone relationship. Additionally, it's implicitly connected with ENT-1744 and ENT-3547 due to the semantic similarities.

3.1.2 Knowledge Graph Construction. Graph construction is delineated into two phases: intra-ticket parsing and inter-ticket connection. **1) Intra-Ticket Parsing Phase:** This phase transforms each text-based ticket t_i into a tree representation $\mathcal{T}_i$. We employ a hybrid methodology, initially utilizing rule-based extraction for predefined fields, such as code sections identified via keywords. Subsequently, for text not amenable to rule-based parsing, we engage an LLM for parsing. The LLM is directed by a YAML template $\mathcal{T}_{\text{template}}$, representing in graph the ticket sections routinely utilized by customer support. **2) Inter-Ticket Connection Phase:** Here, individual trees $\mathcal{T}_i$ are amalgamated into a comprehensive graph $\mathcal{G}$. Explicit connections $\mathcal{E}_{\text{exp}}$ are delineated as specified within tickets, exemplified by designated fields in Jira [2]. Implicit connections $\mathcal{E}_{\text{imp}}$ are inferred from textual-semantic similarities across ticket titles, employing embedding techniques and a threshold mechanism to discern the most relevant tickets for each issue ticket.

$$t_i = t_{i,\text{rule}} \cup t_{i,\text{llm}}$$

$$\mathcal{T}_i = \text{RuleParse}(t_{i,\text{rule}}) + \text{LLMParse}(t_{i,\text{llm}}, \mathcal{T}_{\text{template}}, \text{prompt})$$

$$\mathcal{E}_{\text{exp}} = \{(\mathcal{T}_i, \mathcal{T}_j) \mid \mathcal{T}_i \text{ explicitly connected to } \mathcal{T}_j\}$$

$$\mathcal{E}_{\text{imp}} = \{(\mathcal{T}_i, \mathcal{T}_j) \mid \cos(\text{embed}(\mathcal{T}_i), \text{embed}(\mathcal{T}_j)) \geq \theta\}$$

3.1.3 Embedding Generation. To support online embedding-based retrieval, we generate embeddings for graph node values using pre-trained text-embedding models like BERT [6] and E5 [17], specifically targeting nodes for text-rich sections such as "issue summary", "issue description", and "steps to reproduce", etc. These embeddings are then stored in a vector database (for instance, QDrant [12]). For most cases the text-length within each node can meet the text-embedding model's context length constraints, but for certain lengthy texts, we can safely divide the text into smaller chunks for individual embedding without worrying about quality since the text all belong to the same section.
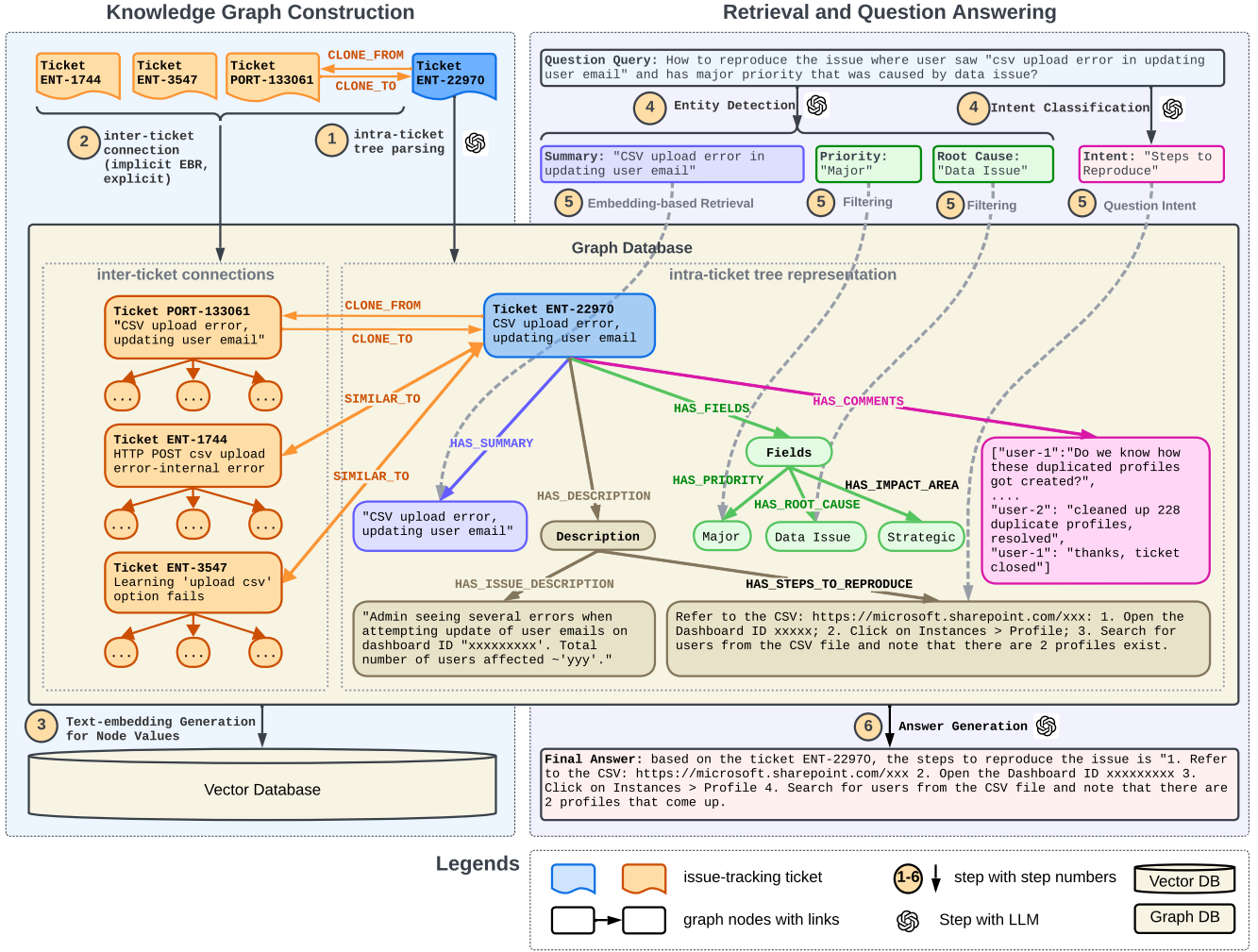
Figure 1: An overview of our proposed retrieval-augmented generation with knowledge graph framework. The left side of this diagram illustrates the knowledge graph construction; the right side shows the retrieval and question answering process.

3.2 Retrieval and Question Answering

3.2.1 Query Entity Identification and Intent Detection.
In this step, we extract the named entities $\mathcal{P}$ of type $\mathrm{Map}(\mathcal{N} \rightarrow \mathcal{V})$ and the query intent set $\mathcal{I}$ from each user query q. The method involves parsing each query q into a key-value pair, where each key n, mentioned within the query, corresponds to an element in the graph template $\mathcal{T}_{\mathrm{template}}$, and the value v represents the information extracted from the query. Concurrently, the query intents $\mathcal{I}$ include the entities mentioned in the graph template $\mathcal{T}_{\mathrm{template}}$ that the query aims to address. We leverage LLM with a suitable prompt in this parsing process. For instance, given the query $q =$ *"How to reproduce the login issue where a user can't log in to LinkedIn?"*, the extracted entity is $\mathcal{P} = \mathrm{Map}(\texttt{"issue summary"} \rightarrow \texttt{"login issue"},$ $\texttt{"issue description"} \rightarrow \texttt{"user can't log in to LinkedIn"})$, and the intent set is $\mathcal{I} = \mathrm{Set}(\texttt{"fix solution"})$. This method demonstrates notable flexibility in accommodating varied query formulations

by leveraging the LLM's extensive understanding and interpretive capabilities.

$$P, I = \mathrm{LLM}(q, T_{\mathrm{template}}, \mathrm{prompt})$$

3.2.2 Embedding-based Retrieval of Sub-graphs.
Our method extracts pertinent sub-graphs from the knowledge graph, aligned with user-provided specifics such as `"issue description"` and `"issue summary"`, as well as user intentions like `"fix solution"`. This process consists of two primary steps: EBR-based ticket identification and LLM-driven subgraph extraction.

In the **EBR-based ticket identification step**, the top K_{ticket} most relevant historical issue tickets are pinpointed by harnessing the named entity set $\mathcal{P}$ derived from user queries. For each entity pair $(k, v) \in \mathcal{P}$, cosine similarity is computed between the entity value v and all graph nodes n corresponding to section k via pretrained text embeddings. Aggregating these node-level scores to ticket-level by summing contributions from nodes belonging to

the same ticket, we rank and select the top K_{ticket} tickets. This method presupposes that the occurrence of multiple query entities is indicative of pertinent links, thus improving retrieval precision.

$$S_{T_i} = \sum_{(k,v) \in \mathcal{P}} \left[\sum_{n \in T_i} \mathbb{I}\{n.\text{sec} = k\} \cdot \cos(\text{embed}(v), \text{embed}(n.\text{text})) \right]$$

In the **LLM-driven subgraph extraction step**, the system first rephrases the original user query q to include the retrieved ticket ID; the modified query q' is then translated into a graph database language, such as `Cypher` for `Neo4j` for question answering. For instance, from the initial query q =*"how to reproduce the issue where user saw 'csv upload error in updating user email' with major priority due to a data issue"*, the query is reformulated to *"how to reproduce 'ENT-22970'"* and thereafter transposed into the Cypher query `MATCH (j:Ticket {ticket_ID: 'ENT-22970'}) -[:HAS_DESCRIPTION]-> (description:Description) -[:HAS_STEPS_TO_REPRODUCE]-> (steps_to_reproduce: StepsToReproduce) RETURN steps_to_reproduce.value`. It is noteworthy that the LLM-driven query formulation is sufficiently versatile to retrieve information across subgraphs, whether they originate from the same tree or distinct trees within the knowledge graph.

3.2.3 Answer Generation. Answers are synthesized by correlating retrieved data from Section 3.2.2 with the initial query. The LLM serves as a decoder to formulate responses to user inquiries given the retrieved information. For robust online serving, if query execution encounters issues, a fallback mechanism reverts to a baseline text-based retrieval method

4 EXPERIMENT

4.1 Experiment Design

Our evaluation employed a curated "golden" dataset comprising typical queries, support tickets, and their authoritative solutions. The control group operated with conventional text-based EBR, while the experimental group applied the methodology outlined in this study. For both groups, we utilized the same LLM, specifically GPT-4 [1], and the same embedding model, E5 [17]. We measured retrieval efficacy using Mean Reciprocal Rank (MRR), recall@K, and NDCG@K. MRR gauges the average inverse rank of the initial correct response, recall@K determines the likelihood of a relevant item's appearance within the top K selections, and NDCG@K appraises the rank quality by considering both position and pertinence of items. For question-answering performance, we juxtaposed the "golden" solutions against the generated responses, utilizing metrics such as BLEU [11], ROUGE [9], and METEOR [3] scores.

4.2 Result and Analysis

The retrieval and question-answering performances are presented in Table 1 and Table 2, respectively. Across all metrics, our method demonstrates consistent improvements. Notably, it surpasses the baseline by 77.6% in MRR and by 0.32 in BLEU score, substantiating its superior retrieval efficacy and question-answering accuracy.

Table 1: Retrieval Performance

	MRR	Recall@K		NDCG@K	
		K=1	K=3	K=1	K=3
Baseline	0.522	0.400	0.640	0.400	0.520
Experiment	**0.927**	**0.860**	**1.000**	**0.860**	**0.946**

Table 2: Question Answering Performance

	BLEU	METEOR	ROUGE
Baseline	0.057	0.279	0.183
Experiment	**0.377**	**0.613**	**0.546**

5 PRODUCTION USE CASE

We deployed our method within LinkedIn's customer service team, covering multiple product lines. The team was split randomly into two groups: one used our system, while the other stuck to traditional manual methods. As shown in Table 3, the group using our system achieved significant gains, reducing the median resolution time per issue by 28.6%. This highlights our system's effectiveness in enhancing customer service efficiency.

Table 3: Customer Support Issue Resolution Time

Group	Mean	P50	P90
Tool Not Used	40 Hours	7 Hours	87 Hours
Tool Used	**15 hours**	**5 hours**	**47 hours**

6 CONCLUSIONS AND FUTURE WORK

In conclusion, our research significantly advances automated question answering systems for customer service. Integrating retrieval augmented generation (RAG) with a knowledge graph (KG) has improved retrieval and answering metrics, and overall service effectiveness. Future work will focus on: developing an automated mechanism for extracting graph templates, enhancing system adaptability; investigating dynamic updates to the knowledge graph based on user queries to improve real-time responsiveness; and exploring the system's applicability in other contexts beyond customer service.

7 COMPANY PORTRAIT

About LinkedIn: Founded in 2003, LinkedIn connects the world's professionals to make them more productive and successful. With more than 1 billion members worldwide, including executives from every Fortune 500 company, LinkedIn is the world's largest professional network. The company has a diversified business model with revenue coming from Talent Solutions, Marketing Solutions, Sales Solutions and Premium Subscriptions products. Headquartered in Silicon Valley, LinkedIn has offices across the globe. Please visit https://www.linkedin.com/company/linkedin/about/ for more information.

8   PRESENTER BIO

Zhentao Xu is a Senior Software Engineer at LinkedIn. He received his M.S. in Robotics and B.S. in Electrical Engineering and Computer Science (EECS) from University of Michigan. His research interests lie in large language models and natural language generation.

REFERENCES

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

[2] Atlassian. 2024. Jira | Issue & Project Tracking Software. https://www.atlassian.com/software/jira Accessed: 2024-01-04.

[3] Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*. 65–72.

[4] Nikita Bhutani, Xinyi Zheng, Kun Qian, Yunyao Li, and H Jagadish. 2020. Answering complex questions by combining information from curated and extracted knowledge bases. In *Proceedings of the first workshop on natural language interfaces*. 1–10.

[5] Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014. Open question answering with weakly supervised embedding models. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part I 14*. Springer, 165–180.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[7] Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. 2023. Large Language Models on Graphs: A Comprehensive Survey. *arXiv preprint arXiv:2312.02783* (2023).

[8] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems* 33 (2020), 9459–9474.

[9] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*. 74–81.

[10] Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2023. Reasoning on graphs: Faithful and interpretable large language model reasoning. *arXiv preprint arXiv:2310.01061* (2023).

[11] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 311–318.

[12] Qdrant Team. 2024. Qdrant - Vector Database. https://qdrant.tech/. Accessed: 2024-01-08.

[13] Zhixiao Qi, Yijiong Yu, Meiqi Tu, Junyi Tan, and Yongfeng Huang. 2023. Foodgpt: A large language model in food testing domain with incremental pre-training and knowledge graph prompt. *arXiv preprint arXiv:2308.10173* (2023).

[14] Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. 2018. Open Domain Question Answering Using Early Fusion of Knowledge Bases and Text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (Eds.). Association for Computational Linguistics, Brussels, Belgium, 4231–4242. https://doi.org/10.18653/v1/D18-1455

[15] Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Heung-Yeung Shum, and Jian Guo. 2023. Think-on-graph: Deep and responsible reasoning of large language model with knowledge graph. *arXiv preprint arXiv:2307.07697* (2023).

[16] Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. 2012. Template-based question answering over RDF data. In *Proceedings of the 21st international conference on World Wide Web*. 639–648.

[17] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533* (2022).

[18] Yilin Wen, Zifeng Wang, and Jimeng Sun. 2023. Mindmap: Knowledge graph prompting sparks graph of thoughts in large language models. *arXiv preprint arXiv:2308.09729* (2023).

[19] Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Hybrid question answering over knowledge base and free text. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. 2397–2407.

[20] Linyao Yang, Hongyang Chen, Zhao Li, Xiao Ding, and Xindong Wu. 2023. ChatGPT is not Enough: Enhancing Large Language Models with Knowledge Graphs for Fact-aware Language Modeling. *arXiv preprint arXiv:2306.11489* (2023).

[21] Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the Joint Conference of the 53rd Annual Meeting of the ACL and the 7th International Joint Conference on Natural Language Processing of the AFNLP*.