Context Matters: Pushing the Boundaries of Open-Ended Answer Generation with Graph-Structured Knowledge Context

Somnath Banerjee [†] **Amruit Sahoo** [†] **Sayan Layek** [†] **Avik Dutta** [†]
Rima Hazra [⊤] **Animesh Mukherjee** [†]

[†]Indian Institute of Technology Kharagpur, India
[⊤]Singapore University of Technology and Design, Singapore
{som.iitkgpcse,sayanlayek2002}@kgpian.iitkgp.ac.in
{rima_hazra}@sutd.edu.sg

Abstract

This paper introduces a novel framework that combines graph-driven context retrieval in conjunction to knowledge graphs based enhancement, honing the proficiency of LLMs, especially in domain specific community question answering platforms like AskUbuntu, Unix, and ServerFault. We conduct experiments on various LLMs with different parameter sizes to evaluate their ability to ground knowledge and determine factual accuracy in answers to open-ended questions. Our methodology GRAPH-CONTEXTGEN consistently outperforms dominant text-based retrieval systems, demonstrating its robustness and adaptability to a larger number of use cases. This advancement highlights the importance of pairing context rich data retrieval with LLMs, offering a renewed approach to knowledge sourcing and generation in AI systems. We also show that, due to rich contextual data retrieval, the crucial entities, along with the generated answer, remain factually coherent with the gold answer.

1 Introduction

In artificial intelligence, Large Language Models (LLMs)(Roberts et al., 2020; Kaplan et al., 2020) have revolutionized text understanding(Lian et al., 2023) and generation (Wei et al., 2023). Despite their impressive capabilities, LLMs struggle in low-resource settings (Chen et al., 2023; Guu et al., 2020), are constrained by knowledge cutoffs, and often produce hallucinations (McKenna et al., 2023). Additionally, managing the trade-off between quality and the vast number of parameters (Xu et al., 2023) presents challenges, particularly for researchers with limited resources.

To overcome these limitations, new methods such as grounding LLMs [1] and Retrieval-Augmented Generation (RAG) (Yu et al., 2023) have been proposed. These approaches enable models to access
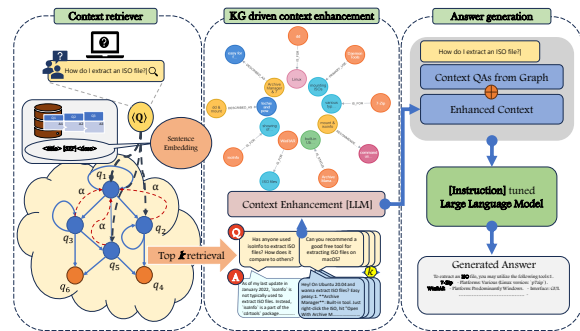

Figure 1: GRAPHCONTEXTGEN framework.

external databases, enhancing their responses with current, detailed, and accurate information.

A critical aspect of effective knowledge grounding is the retrieval mechanism (Lewis et al., 2021). Traditional text-based retrieval methods are evolving to handle more complex questions, moving beyond simple keyword matching. Current techniques often struggle with determining optimal chunk sizes [2] for indexing and querying, leading to inconsistent results. Graph-based retrieval systems offer a solution by capturing intricate relationships through structured data, providing deeper semantic understanding and more contextually relevant results (Zhang et al., 2021). These systems adapt to evolving data, uncovering insights and forming connections among diverse entities.

This technique is vital in various applications, including dialogue systems (Li et al., 2022), open-domain question answering (Lu et al., 2023), and novelty-controlled paraphrasing (Xie et al., 2023). For example, StackOverflow's OverflowAI [3] aims to refine search and enhance code and discussion platforms. Automated answer generation on community Q&A platforms promises timely and accurate information, reducing errors and providing immediate knowledge access. Unlike past research,

---

[1]https://techcommunity.microsoft.com/t5/fasttrack-for-azure/grounding-llms/ba-p/3843857

[2]https://www.pinecone.io/learn/chunking-strategies/
[3]https://stackoverflow.blog/2023/07/27/announcing-overflowai/

our study uniquely employs LLMs to generate tailored answers for these platforms.

We introduce the GRAPHCONTEXTGEN framework, which combines graph-based retrieval with LLMs to enhance context and ensure factual accuracy. Our extensive experiments in low-resource domains like AskUbuntu[4], Unix [5], and ServerFault [6] demonstrate the effectiveness and resilience of LLM-generated answers, even in specialized areas.

Contribution: The key contribution of this paper is as follows.

- We introduce GRAPHCONTEXTGEN, a framework that integrates graph-based retrieval with context enhancement using a knowledge graph for CQA answer generation. This approach consistently outperforms previous SOTA methods. Additionally, instruction tuning with LLMs further improves performance (see Section 4).

- We evaluate a range of recently released LLMs from 1.5B to 40B on CQA of low resource domains for answer generation in a zero shot setting (see Table 2).

- In addition to automatic evaluation, we also perform evaluation based on human judgements and demonstrate that in both cases our proposed framework consistently outperforms all current SOTA text-based retrieval techniques (see Table 3 and Section 6).

- We conduct a detailed retrospective analysis to compare actual answers with those generated by our framework, focusing on their factual alignment. The generated answers typically align well with the actual ones (see Figure 2).

2 Related Work

Over the years several approaches such as feature based methods (Wang et al., 2009; Wang and Manning, 2010), CNN (Severyn and Moschitti, 2015; Rao et al., 2017), RNN (Wang and

---

[4]https://askubuntu.com/
[5]https://community.unix.com/
[6]https://serverfault.com/

Nyberg, 2015), attention mechanism (Tan et al., 2016; dos Santos et al., 2016) have been proposed for answer selection and summarization. Some recent research focuses on summarizing diverse content on StackOverflow (Chengran et al., 2022), using AnswerBot–an answer summary generator (Xu et al., 2017), Opiner–which summarizes API reviews (Uddin and Khomh, 2017), extracting key sentences to guide developers on StackOverflow (Nadi and Treude, 2019), and multi-document summarization (Xu and Lapata, 2020). In the realm of answer summarization, numerous studies (Ganesan et al., 2010; Naghshzan et al., 2021) harness graphical structures, leverage existing graph-based summarizers (Mihalcea and Tarau, 2004; Erkan and Radev, 2004; Kazemi et al., 2020), and employ graph-centric measures. In the age of LLMs, some research has centered around controlled summary generation via effective keyword-based prompting (He et al., 2022).

3 Dataset

Attributes	AskUbuntu		Unix		Serverfault	
	Train	Test	Train	Test	Train	Test
Size	15,505	203	19,742	241	10908	226
Year of questions	2019-20	2021-23	2019-20	2021-23	2019-20	2021-23
Avg. length of questions	254.38	156.65	205.85	220.57	259.81	248.93
Avg. length of answers	122.22	217.16	181.17	210.02	145.30	161.73

Table 1: Dataset statistics.

In this experiment, we select three domain-specific datasets from open-source CQA platforms: AskUbuntu, ServerFault, and Unix, all of which originate from a low-resource domain with minimal properly annotated data available on these topics. These datasets, considered from June 2023, includes questions (title and body), a list of answers, an accepted answer flag, tags for the questions, and the posting dates and times for both questions and answers. For each question, the accepted answer serves as the ground truth. Due to the limited resources of the datasets and the high expenses associated with human involvement, we opt not to use human annotations. Further we apply several filtering procedures, such as duplicate question removal, non-specific answer removal, and length constraints (token limit in LLM) resulting in our dataset. Adopting the temporal splitting approach inspired by (Hazra et al., 2021, 2023), we consider training set from 2019-2020 and test dataset from 2021-2023. Due to resource limitations, we randomly sample test dataset, the details of which are provided in Table 1. From the training

set of each dataset, we construct an instruction-tuning dataset by pairing questions and answers in the format '$[INST]Question[\backslash INST]Answer : actual\_answer$'. We prepare this instruction data for each dataset.

4 Methodology

In this section, we explain our proposed framework GRAPHCONTEXTGEN. The overall framework is shown in Figure 1. Our proposed framework consists of three modules − (1) context retriever (2) KG-driven context enhancement (3) answer generation. Before explaining every module, we describe the problem in detail below.

4.1 Preliminaries

Given a community question answering (CQA) system, there is a collection of questions and their associated accepted answers, represented as $< Q, A_Q > = \{(q_1, a_{q_1}), (q_2, a_{q_2}), ... (q_n, a_{q_n})\}$. The anchor question (query) is represented by q. We consider subset of question pool Q_{pool}, where $Q_{pool} \subset Q$. We represent our instruction tuned dataset as $\mathcal{D_{INST}}$ which contains instruction $\mathcal{INST}$, question pool Q_{pool} and their accepted answer $A_{Q_{pool}}$. Our objective is formalized as follows.

Context retriever: In this module, we consider

Algorithm 1 GRAPHCONTEXTGEN

1: Input: Initial question pool Q_{pool}, query q, LLM M, instruction dataset $\mathcal{D_{INST}}$
2: **function** RETRIEVER(Q_{pool}, q)
3:    Build $G(V, E)$, nodes $V = Q_{pool}$, edges $E \subseteq Q_{pool} \times Q_{pool}$ where $sim(q_a, q_b) > T \ \forall \ q_a, q_b \in Q_{pool}$
4:    Build extended graph $G'(V', E')$ where $V' = V \cup q$ and $E' = E \cup E_q$
5:    Q_{pool}^{ranked} = sort($QueryAwarePageRank(G')$)
6:    Choose a set of top k questions $Q_{top_k}^q$
7: **end function**
8: **function** CONTEXTENHANCER($Q_{top_k}^q, q$)
9:    context $C^q = < Q_{top_k}^q, A_{Q_{top_k}^q} >$
10:    Extract $\tau^{init}(h, r, t)$ using LLM M and REBEL from C^q
11:    $Ent(C^q) = EntitySetBuilder(\tau^{init})$ where $Ent(C^q)$ contains set of entities $e_1, e_2, ..., e_n$
12:    Extract triplets $\tau(h, r, t)$ from Wikidata for $h \in Ent(C^q)$
13:    Filtered triples set τ' if $t \in Ent(C^q)$
14:    Prepare triplets set $\tau^f = \tau^{init} \cup \tau'$
15:    Build sequence of sentences S from all triplets τ^f
16:    Enhanced context $C_{enc}^q = C_q \oplus S$
17: **end function**
18: **function** ANSWERGENERATOR(C_{enc}^q, q)
19:    $M' = SupervisedFineTuning(M, \mathcal{D_{INST}})$
20:    $a_q^{gen} = M'(C_{enc}^q, q)$
21: **end function**

---

anchor question q and the Q_{pool} as input and output the most relevant questions from the Q_{pool}. The set of relevant questions is represented by $Q_{top_k}^q$. We explain the working procedure of the module in subsequent sections.

KG driven context enhancement: This module takes the query q and the final set of most relevant question $Q_{top_k}^q$ as input to formulate enhanced context. The initial context is represented by C^q which is the $< Q_{top_k}^q, A_{Q_{top_k}^q} >$ pairs. Further, we represent the sequence of sentences by S that are obtained from the entity extraction procedure and knowledge graph. Enhanced context is represented by C_{enc}^q.

Answer generation: In this module, we provide the query q and enhanced context C_{enc}^q as input to generate the answer denoted by a_q^{gen}. We denote the ground truth answer as a_q^{gt}. We explain each of the above-mentioned steps in subsequent sections.

4.2 Context retriever

The objective of this module is to retrieve relevant previous questions given the query question. Our $RETRIEVER$ module in Algorithm 1, consists of two parts − (I) question-question graph (Q-Q graph) construction and (II) retrieval of top relevant questions.

(I) Q-Q graph construction: We build a question-question graph (Q-Q graph) to obtain the relevant questions from the previously posted question pool Q_{pool}. In a Q-Q graph ($G(V, E)$), nodes (V) are the questions and the edges (E) are formed based on the cosine similarity between the concatenated embeddings of the title and the body of two questions. We include the edge only if the similarity score crosses a particular threshold[7]. The major motivation for building the Q-Q graph is that it can help to identify semantically similar questions based on the structural properties of the graph. This systematically prepared graph will be utilized to prioritize a set of existing questions given a query q.

(II) Retrieval of top relevant questions: For a given query q, we extend the existing Q-Q graph $G(V, E)$ to $G'(V', E')$. We form the graph G' by including the query q as a node and further measure the similarity with all the nodes in G. If the similarity score passes a threshold[8], the edges (E_q) are formed between question q to the respective nodes in G accordingly. We conceptualize that questions (in graph G') with high node centric score from the perspective of the query node q could be considered as the relevant questions (nodes) to the query q.

---

[7]Empirically identified based on graph density.
[8]We followed the same threshold used in Q-Q graph construction.

We use personalized PageRank (PPR) (Wang et al., 2020) which introduces bias toward the query node and tailor the ranking based on the node preference (i.e., prior information).

We obtain PPR scores for all the nodes (except q) in graph G'. For the given query node q, we select the top k relevant questions. This top k question set is referred to as $Q_{top_k}^q$. We do the above mentioned process for all the queries in the query set.

4.3 KG driven context enhancement

From the previous module, we obtain $Q_{top_k}^q$ questions and their answers $A_{Q_{top_k}^q}$ and use them as context C^q for a query q. It is observed that LLMs lack in generating aligned answers for open ended questions (Ai et al., 2023) even after providing the relevant context. In this module, we attempt to enhance the retrieved context C^q. In this process (see CONTEXTENHANCER module in algorithm 1), we follow two major steps – (i) entity identification and triplet formation, (ii) enhanced context formulation.

Entity identification and triplet formation: In this stage, we first identify all the important information (e.g., entities) present in the C^q. For important information identification, we employ the LLM M and REBEL (Huguet Cabot and Navigli, 2021) to obtain initial relation triplets (τ^{init}) from the context C^q. We use a simple prompt plus the context C^q to the LLM M for relation triplet extraction task. In case of REBEL, we obtain the triplets by passing C^q as input to their internal function. Note that a triplet consists of (head_entity, tail_entity, relation). We prepare a set $Ent(C^q)$ which contains all the entities present in these triplets.Further, we use Wikidata to obtain one hop neighbors of each entity and their relationship again in the form of triplets. We now consider all the new triplets (τ') as well as those in τ^{init} to prepare a new extended set of triplets τ^f. We retain only those triplets in τ' whose head_entity and tail_entity are present in the original context C^q. **Enhanced context formulation**: We construct a set of sub-contexts (S) in the form of sequence of sentences from triplet set τ^f. Basically, we form the sentence by placing the head entity, the relation and the tail entity in sequence.We finally construct the enhanced context C_{enc}^q by concatenating the actual context C_q and S. We illustrate the process in Figure 4.

4.4 Answer generation

In this section, we use the enhanced context C_{enc}^q and the given query q to generate the answers using LLM. In this component, we use the LLM in two ways – pretrained LLM and finetuned LLM. In the pretrained setup, we pass the enhanced context C_{enc}^q and the query question q as input and obtain the answer as output. In this setting, we use the LLM (model M) as black box. For fine tuned version, we utilize instruction dataset $\mathcal{D}_{\mathcal{INST}}$ to efficiently fine tune the LLM M. The fine tuned model is represented as M' Further, we use the enhanced context C_{enc}^q and query q as input to the fine tuned model M' and obtain the generated answer a_q^{gen}.

5 Experimental setup

Baselines: In this work, we use various methods as baselines. Some of the baselines are proposed by us which we believe are very competitive to our best approach.
Pre-LLM era baselines: We compare our approach with SOTA answer generation/summarization works such as **AnswerBot** (Xu et al., 2017; Cai et al., 2019), **GenQA** (Hsu et al., 2021) and **TechSumBot** (Yang et al., 2023a) (see Appendix A.2). Due to unavailability of the codebase and unclear implementation details, we could not compare this paper (Deng et al., 2019) with our method. **Zeroshot LLMs**: In this setting, we use various competitive LLMs to generate the answer of the given question. The LLMs are of different parameter sizes (7B to 40B). Such a choice enables us to understand how well models with diverse parameter sizes perform in zero shot setting. **[w/o INST] TEXTGEN**: In this setup, we use a vector database (chromaDB[9] and FAISS[10]) containing all the training set questions. We compute contextual similarity between query q and all the questions in database. We rank the questions in database based on the cosine similarity scores (higher scores get top ranks) and retrieve top k questions. Further we use the top k questions and their actual answers as few shot examples to the pretrained LLM for generating the answer. **[w/o INST] TEXTCONTEXTGEN**: In this setup, we retrieve the top k questions and their answers using the same method as **[w/o INST] TEXTGEN**. Subsequently, we use

---

[9]https://docs.trychroma.com/getting-started
[10]https://python.langchain.com/docs/integrations/vectorstores/faiss

Method	AskUbuntu			Unix			ServerFault		
Size ↑	BERTScore	ROUGE 1	ROUGE L	BERTScore	ROUGE 1	ROUGE L	BERTScore	ROUGE 1	ROUGE L
	macro-F1 score								
Phi (1.5B) (Li et al., 2023)	0.803	0.219	0.202	0.792	0.191	0.176	0.790	0.202	0.183
Falcon (7B) (fal)	0.718	0.167	0.153	0.794	0.151	0.138	0.801	0.181	0.166
MPT (7B) (Team, 2023)	0.738	0.156	0.147	0.786	0.138	0.127	0.709	0.144	0.132
StackLlama (7B) (Beeching et al., 2023)	0.797	0.136	0.130	0.774	0.122	0.112	0.785	0.131	0.120
Llama2 (7B) (Touvron et al., 2023)	0.809	0.221	0.204	0.792	0.178	0.163	0.813	0.183	0.167
Flan-t5-xxl (11B) (Roberts et al., 2022)	0.795	0.131	0.114	0.792	0.106	0.098	0.816	0.135	0.124
Vicuna (13B) (Chiang et al., 2023)	0.741	0.177	0.163	0.789	0.181	0.165	0.810	0.185	0.169
Llama2 (13B) (Touvron et al., 2023)	0.810	0.227	0.211	0.806	0.191	0.175	0.819	0.189	0.176
Gpt-neox (20B) (Black et al., 2022)	0.724	0.136	0.127	0.776	0.140	0.131	0.799	0.152	0.161
Falcon (40B) (fal)	0.721	0.182	0.167	0.801	0.179	0.171	0.812	0.186	0.173

Table 2: Comparison of zero-shot learning performance of various models of different sizes across the AskUbuntu, Unix, and ServerFault datasets. Metrics include BERTScore, ROUGE 1, and ROUGE L scores. The cell color intensity indicates the relative performance, with darker shades representing higher values. The best results are marked with the darkest shade of cyan for BERTScore & magenta for ROUGE scores.

Method	AskUbuntu				Unix				ServerFault			
	BERTScore	ROUGE 1	ROUGE L	FactSumm	BERTScore	ROUGE 1	ROUGE L	FactSumm	BERTScore	ROUGE 1	ROUGE L	FactSumm
	macro-F1 score				macro-F1 score				macro-F1 score			
Pre-LLM era												
AnswerBot (Xu et al., 2017)	0.803	0.236	0.111	0.578	0.791	0.191	0.091	0.583	0.802	0.191	0.094	0.642
GenQA (Hsu et al., 2021)	0.781	0.095	0.071	0.551	0.55	0.048	0.04	0.427	0.668	0.059	0.045	0.662
TechSumBot (Yang et al., 2023b)	0.781	0.100	0.05	0.580	0.776	0.077	0.039	0.563	0.781	0.064	0.034	0.655
LLM era (best performing LLM from Table 2 is used, i.e., Llama2 (13B))												
[w/o INST] TEXTGEN	0.812	0.217	0.202	0.612	0.809	0.179	0.162	0.683	0.810	0.179	0.166	0.733
[w/o INST] TEXTCONTEXTGEN	0.827	0.223	0.204	0.619	0.818	0.184	0.168	0.683	0.823	0.198	0.175	0.738
[w/o INST] GRAPHGEN	0.823	0.204	0.188	0.619	0.809	0.181	0.162	0.683	0.816	0.182	0.166	0.737
[w/o INST] GRAPHCONTEXTGEN	0.831	0.222	0.206	0.621	0.822	0.184	0.169	0.685	0.823	0.197	0.175	0.738
FineTuned GEN Zero-Shot	0.815	0.203	0.187	0.608	0.812	0.183	0.167	0.661	0.821	0.195	0.179	0.733
TEXTGEN	0.821	0.183	0.170	0.623	0.823	0.186	0.169	0.684	0.829	0.197	0.179	0.738
TEXTCONTEXTGEN	0.833	0.221	0.200	0.636	0.834	0.182	0.161	0.689	0.831	0.198	0.180	0.739
GRAPHGEN	0.827	0.182	0.170	0.636	0.817	0.183	0.164	0.691	0.831	0.198	0.180	0.737
GRAPHCONTEXTGEN*	0.840	0.214	0.189	0.639	0.837	0.187	0.169	0.693	0.839	0.198	0.181	0.737

Table 3: Comparison of various question-answering and summarization methods on AskUbuntu, Unix, and ServerFault platforms using evaluation metrics BERTScore, ROUGE 1, ROUGE L, and FactSumm. Methods are categorized into those developed before the LLM era (pre-LLM era) and those developed during the LLM era. * p-value < 0.05 on comparison with pre-LLM era models. The best results are marked with the darkest shade of cyan for BERTScore, magenta for ROUGE score & blue for FactSumm.

context enhancement component of our approach to enhance the context. Further we provide the enhanced context and the query as input to the pretrained LLM and obtain the generated answer. **[w/o INST] GRAPHGEN**: In this setup, we use the RETRIEVER module of our algorithm to retrieve the top k questions. We use k questions and their answers as few shot examples to the pretrained LLM for generating the answer. **[w/o INST] GRAPHCONTEXTGEN**: We follow the retrieval step from **[w/o INST] GRAPHGEN**. Further we use our CONTEXTENHANCER module to enhance the context. **FINETUNED GEN ZERO-SHOT**: In this setting, we use instruction fine tuned LLM in zero shot settings. Here, we pass the questions as input and the fine tuned LLM generates the answer. **TEXTGEN**: This setup is same as **[w/o INST] TEXTGEN**. However, we use our instruction fine tuned LLM for generation. **GRAPHGEN**: This setup is same as **[w/o INST] GRAPHGEN**. However, we use our instruction fine tuned LLM for generation.

Parameter setting[11]: In our method GRAPHCON-

TEXTGEN, we use Flag embedding (Xiao et al., 2023) (bge-large-en) to obtain embedding for each question in the training set. The dimension of the embedding is 1024. We construct the edges of the Q-Q graph if the embedding cosine similarity between two questions cross a threshold of 0.8 [12]. In PPR algorithm, the α value is set to 0.85, max_iter is set to 100 and tol is set to 1e-6. The k value is set to 2 [13]. For parameter settings of instruction tuned models, see Appendix A.1.

Evaluation metrics: We have used three metrics – ROUGE score[14], BERT score (Zhang* et al., 2020; Zhang et al., 2020) and FactSumm score (Heo, 2021) for automatic evaluation of generated answers. Note that the FactSumm (Heo, 2021) package extracts the facts from the generated text and the ground truth text and computes an overall score based on the fact overlap and fact mismatch. This package has also been used in earlier works (Liu et al., 2021; Qian et al., 2023) to measure the factual accuracy of the generated text.

---

[11]Values of all these hyperparameters are obtained through grid search.

[12]Empirically computed based on graph density.

[13]Empirically identified to fit the whole context within the acceptable token limit of the LLMs.

[14]https://huggingface.co/spaces/evaluate-metric/rouge

6 Results

The Table 2 notes the BERTScore, ROUGE 1, ROUGE L (macro-F1) values achieved by different LLMs in a zero-shot setup across three domains – AskUbuntu, Unix, and ServerFault.

Zero-shot answer generation by different LLMs: For AskUbuntu, Llama2 (13B) achieves the highest BERTScore (0.810), ROUGE 1 (0.227), and ROUGE L (0.211). For Unix, Llama2 (13B) leads in BERTScore (0.806), while Vicuna (13B) tops ROUGE 1 (0.181), and Llama2 (13B) tops ROUGE L (0.175). For ServerFault, Llama2 (13B) dominates BERTScore (0.819), Vicuna (13B) leads in ROUGE 1 (0.185), and Llama2 (13B) in ROUGE L (0.176). Performances are not always proportional to model size, as Llama2 (13B) often outperforms larger models like Gpt-neox (20B) and Falcon (40B). Models of similar sizes also display varied performances, indicating the importance of architecture and training methods.

Main results: Table 3 compares baseline results with our proposed method. **Pre-LLM era:** AnswerBot shows competitive BERTScore (0.803, 0.791, 0.802) for AskUbuntu, Unix, and ServerFault, respectively, while GenQA underperforms on Unix (BERTScore 0.55). AnswerBot achieves the highest FactSumm score across all platforms. **LLM era:** Llama2 (13B) is the reference model for generating answers. Our model, GRAPHCONTEXTGEN, outperforms all baselines in BERTScore and FactSumm for AskUbuntu and Unix, producing factually more correct answers. For ROUGE 1 and ROUGE L, GRAPHCONTEXTGEN shows competitive performance in Unix and ServerFault. Models from the LLM era generally outperform pre-LLM models in these metrics.

Grounding of the generated answers: We use UniversalNER (Zhou et al., 2023) to identify entities in the ground truth and generated answers. The Jaccard similarity between entity sets for our model is 0.85, 0.75, and 0.79 for AskUbuntu, Unix, and ServerFault, respectively (Figure 2(A)). The overlap in the number of triplets is shown in Figure 2(B), indicating our model's answers are rich in entities and relationships present in the ground truth.

6.1 Ablation study

In this section, we attempt to understand how well each component of our model is working and contributing to the overall performance. Here, we have
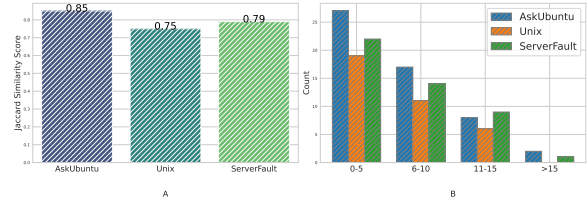

Figure 2: Performance metrics on community datasets using the UniNER model. (A) Jaccard similarity scores illustrate the level of overlap between predicted entities and actual entities. (B) Triplet overlap distribution across different ranges, provide insights into the depth of entity matching in the model's predictions.
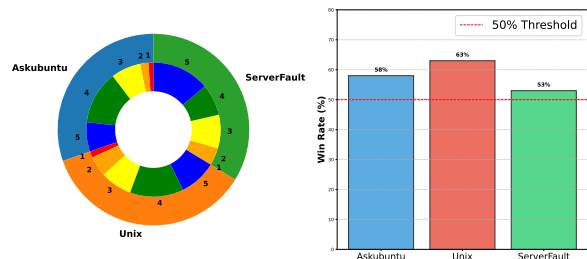

Figure 3: Comparative analysis of human feedback on the generated answers for test questions drawn from Askubuntu, Unix, and ServerFault. On the left, a dual-layer pie chart breaks down the total number of answers and their respective ratings from 1 to 5. The right side displays a bar graph indicating the percentage of wins for answers from each community, benchmarked against a 50% threshold. Notably, a majority of the ratings lean toward the higher end, indicating overall positive reception.

done this study mainly from three different angles (see Table 4).

- *Based on embedding algorithms*: The performance differences between sentence BERT (Reimers and Gurevych, 2019) and BGE embeddings (Xiao et al., 2023) within the TEXTGEN model are evident. The BGE variant demonstrates a slightly improved performance over sentence BERT.

- *Based on one-shot*: When focusing on one-shot methodologies, the GRAPHGEN model achieves marginally better scores compared to the TEXTGEN model. This suggests that the incorporation of graph structures potentially aids in better knowledge retrieval

- *Based on only knowledge graph context*: Focusing solely on the knowledge graph context, two variants emerge: ONLYCON-TEXTGEN[FT] (from text) and ONLYCON-TEXTGEN[FG] (from graph). The 'From Graph' variant consistently exhibits slightly

Method	AskUbuntu	Unix	ServerFault
Based on embedding algorithms			
[w/o INST] TEXTGEN (Sentence BERT)	0.808	0.783	0.803
[w/o INST] TEXTGEN (BGE)	0.812	0.809	0.810
Based on one-shot			
[w/o INST] TEXTGEN	0.809	0.807	0.807
[w/o INST] GRAPHGEN	0.818	0.809	0.812
Based on only knowledge graph context			
[w/o INST] ONLYCONTEXTGEN[FT]	0.819	0.807	0.819
[w/o INST] ONLYCONTEXTGEN[FG]	0.827	0.820	0.818

Table 4: Ablation study highlighting the performance of various methods. FT: From text, FG: From graph.

higher scores across all platforms than its 'From Text' counterpart. This underscores the importance of leveraging structured graph data for enhanced performance in our application.

7 Human evaluation

In the results of the automatic evaluation, it is essential to remember that low values of BERTScore or FactSumm might also correspond to lower-quality ground truth answers posted by humans. Thus, there is a possibility that the model-generated answers are superior in quality compared to the ground truth answers. Such points can be verified only by human judgment experiments presented in Figure 3 and the Appendix A.6.

8 Error Analysis

This section presents a systematic error analysis highlighting the error types and corresponding examples (see Table 7).

Misaligned retrieval outcomes: This misalignment occurs when the retrieved content, accurate in its context, doesn't match the user's intended query, leading to off-target responses due to the generation's reliance on the retrieved data. On platforms like AskUbuntu, Unix, and ServerFault, overlapping themes, like a 'boot issues' query retrieving 'USB booting' content instead of 'system booting problems', exacerbates the issue.

Entity misalignment: This issue arises when the retrieval mechanism accurately finds data but incorrectly links it to an entity in the knowledge graph, causing responses to deviate from the user's context. For example, in UNIX, a term like 'read' might refer to a command or a configuration file, leading to misassociations if not accurately disambiguated.

Composite query conundrums: This problem occurs when a user's query involves multiple issues, and the retrieval system typically focuses primarily on one, neglecting the others. For example, on

platforms like AskUbuntu, a user might ask about 'memory and CPU usage', and the system might only address the memory part.

Factual fidelity fallacies: This issue arises when the RAG system, skilled in retrieval and generation, delivers answers that lack factual accuracy or are outdated, a common issue in rapidly evolving platforms like AskUbuntu. For instance, a query about a software tool may elicit a response based on outdated versions.

Contextual content crux: When the system retrieves broad or limited contents, it can produce answers that lack depth or specificity, a challenge often seen on platforms like AskUbuntu. For example, a query about a specific Ubuntu feature might get a general response if the corpus lacks in-depth content.

9 Complexity analysis

As the dimension is fixed, the cosine similarity between two embeddings is $\mathcal{O}(1)$. For training set questions (say n_q = variable), it becomes $n_q \times \mathcal{O}(1) = \mathcal{O}(n_q)$. By including the query in the graph, pagerank becomes $\mathcal{O}((n_q+1)+(e_q+e')) = \mathcal{O}(n_q+1+e_q+e') = \mathcal{O}(n_q+e_q+e')$. Assuming e' can be at $max(n_q)$, it simplifies to $\mathcal{O}(n_q+n_q+e_q) = \mathcal{O}(n_q + e_q)$. Therefore, the total complexity until pageRank calculation for each test instance is $\mathcal{O}(n_q + e_q)$. The worst-case complexity could be $\mathcal{O}(n_q + n_q^2) = \mathcal{O}(n_q^2)$. For each dataset, n_q and e_q remain constant for all test instances.

10 Conclusion

This study addresses a notable challenge in CQA platforms: the automatic generation of answers across popular platforms often lacks clear problem definitions, leading to issues with proper knowledge grounding and factually incoherent responses. We present GRAPHCONTEXTGEN, which, to the best of our knowledge, is the first solution that uses graph retrieval combined with knowledge graph context for this challenge in domain-specific CQA platforms. Our evaluations indicate that this model outperforms previous prominent approaches. Notably, human evaluators determine that answers generated by GRAPHCONTEXTGEN exhibit greater factual coherence and knowledge grounding. We further demonstrate that researchers with constrained GPU resources can adopt this solution with smaller parameter LLMs and achieve performance that are at par with larger models.

11 Limitation

Using knowledge graphs with LLMs to generate technical answers encounters several challenges. The effectiveness of this approach partially depends on the accuracy and completeness of the knowledge graphs, as any gaps or inaccuracies can mislead the LLM's responses. LLMs tailored for summarization may sacrifice detail for conciseness, potentially overlooking critical nuances of technical topics. Challenges in handling ambiguous contexts and the potential for biases introduced during knowledge graph construction and LLM training further complicate accurate answer generation. Moreover, the computational resources required for processing large-scale knowledge graphs pose scalability issues. Ensuring that knowledge graphs remain up-to-date and that LLMs can adapt to new information without frequent retraining are ongoing concerns. Finally, while LLMs are versatile, they may not achieve the level of specificity and accuracy provided by systems specialized in particular non niche domains.

12 Ethical consideration

The information in our dataset is free from harmful or offensive materials. We take serious measures to anonymize and handle any personal or sensitive data with the highest level of confidentiality. Protection of participants' privacy is our priority and we consistently ensure to acquire their informed consent when collecting, annotating, and analyzing data. We provide equal incentives to all annotators for their efforts toward the annotation task.

References

Technology innovation institute tii. falcon llm, 2023.

Qingyao Ai, Ting Bai, Zhao Cao, Yi Chang, Jiawei Chen, Zhumin Chen, Zhiyong Cheng, Shoubin Dong, Zhicheng Dou, Fuli Feng, Shen Gao, Jiafeng Guo, Xiangnan He, Yanyan Lan, Chenliang Li, Yiqun Liu, Ziyu Lyu, Weizhi Ma, Jun Ma, Zhaochun Ren, Pengjie Ren, Zhiqiang Wang, Mingwen Wang, Ji-Rong Wen, Le Wu, Xin Xin, Jun Xu, Dawei Yin, Peng Zhang, Fan Zhang, Weinan Zhang, Min Zhang, and Xiaofei Zhu. 2023. Information retrieval meets large language models: A strategic report from chinese ir community. *AI Open*, 4:80–90.

Edward Beeching, Younes Belkada, Kashif Rasul, Lewis Tunstall, Leandro von Werra, Nazneen Rajani, and Nathan Lambert. 2023. Stackllama: An rl fine-tuned llama model for stack exchange question and answering.

Sidney Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, Usvsn Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. 2022. GPT-NeoX-20B: An open-source autoregressive language model. In *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 95–136, virtual+Dublin. Association for Computational Linguistics.

Liang Cai, Haoye Wang, Bowen Xu, Qiao Huang, Xin Xia, David Lo, and Zhenchang Xing. 2019. Answerbot: An answer summary generation tool based on stack overflow. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2019, page 1134–1138. Association for Computing Machinery.

Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, and Jiliang Tang. 2023. Exploring the potential of large language models (llms) in learning on graphs.

Yang Chengran, Bowen Xu, Ferdian Thung, Yucen Shi, Ting Zhang, Zhou Yang, Xin Zhou, Jieke Shi, Junda He, DongGyun Han, and David Lo. 2022. Answer summarization for technical queries: Benchmark and new approach.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Yang Deng, Wai Lam, Yuexiang Xie, Daoyuan Chen, Yaliang Li, Min Yang, and Ying Shen. 2019. Joint learning of answer selection and answer summary generation in community question answering.

Cícero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *ArXiv*, abs/1602.03609.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.*, 22(1):457–479.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd international conference on computational linguistics*, pages 340–348. Association for Computational Linguistics.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training.

Rima Hazra, Hardik Aggarwal, Pawan Goyal, Animesh Mukherjee, and Soumen Chakrabarti. 2021. Joint autoregressive and graph models for software and developer social networks. In *Advances in Information Retrieval - 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 - April 1, 2021, Proceedings, Part I*, volume 12656 of *Lecture Notes in Computer Science*, pages 224–237. Springer.

Rima Hazra, Arpit Dwivedi, and Animesh Mukherjee. 2023. Is this bug severe? a¬†text-cum-graph based model for¬†bug severity prediction. In *Machine Learning and Knowledge Discovery in Databases*, pages 236–252, Cham. Springer Nature Switzerland.

Junxian He, Wojciech Kryscinski, Bryan McCann, Nazneen Rajani, and Caiming Xiong. 2022. CTRLsum: Towards generic controllable text summarization. pages 5879–5915, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Hoon Heo. 2021. Factsumm: Factual consistency scorer for abstractive summarization. https://github.com/Huffon/factsumm.

Chao-Chun Hsu, Eric Lind, Luca Soldaini, and Alessandro Moschitti. 2021. Answer generation for retrieval-based question answering systems. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4276–4282, Online. Association for Computational Linguistics.

Pere-Lluís Huguet Cabot and Roberto Navigli. 2021. REBEL: Relation extraction by end-to-end language generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2370–2381, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models.

Ashkan Kazemi, Verónica Pérez-Rosas, and Rada Mihalcea. 2020. Biased TextRank: Unsupervised graph-based content extraction. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1642–1652, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks.

Yu Li, Baolin Peng, Yelong Shen, Yi Mao, Lars Liden, Zhou Yu, and Jianfeng Gao. 2022. Knowledge-grounded dialogue generation with a unified knowledge representation.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. 2023. Textbooks are all you need ii: phi-1.5 technical report.

Long Lian, Boyi Li, Adam Yala, and Trevor Darrell. 2023. Llm-grounded diffusion: Enhancing prompt understanding of text-to-image diffusion models with large language models.

Wei Liu, Huanqin Wu, Wenjing Mu, Zhen Li, Tao Chen, and Dan Nie. 2021. Co2sum: Contrastive learning for factual-consistent abstractive summarization. *ArXiv*, abs/2112.01147.

Yujie Lu, Siqi Ouyang, and Kairui Zhou. 2023. Structured knowledge grounding for question answering.

Nick McKenna, Tianyi Li, Liang Cheng, Mohammad Javad Hosseini, Mark Johnson, and Mark Steedman. 2023. Sources of hallucination by large language models on inference tasks.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

Sarah Nadi and Christoph Treude. 2019. Essential sentences for navigating stack overflow answers.

AmirHossein Naghshzan, Latifa Guerrouj, and Olga Baysal. 2021. Leveraging unsupervised learning to summarize APIs discussed in stack overflow. IEEE.

Hongjing Qian, Yutao Zhu, Zhicheng Dou, Haoqi Gu, Xinyu Zhang, Zheng Liu, Ruofei Lai, Zhao Cao, Jian-Yun Nie, and Ji-Rong Wen. 2023. Webbrain: Learning to generate factually correct articles for queries by grounding on large web corpus.

Jinfeng Rao, Hua He, and Jimmy Lin. 2017. Experiments with convolutional neural network models for answer selection. SIGIR '17, page 1217–1220, New York, NY, USA. Association for Computing Machinery.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks.

Adam Roberts, Hyung Won Chung, Anselm Levskaya, Gaurav Mishra, James Bradbury, Daniel Andor, Sharan Narang, Brian Lester, Colin Gaffney, Afroz Mohiuddin, Curtis Hawthorne, Aitor Lewkowycz, Alex Salcianu, Marc van Zee, Jacob Austin, Sebastian Goodman, Livio Baldini Soares, Haitang Hu, Sasha Tsvyashchenko, Aakanksha Chowdhery, Jasmijn Bastings, Jannis Bulian, Xavier Garcia, Jianmo Ni, Andrew Chen, Kathleen Kenealy, Jonathan H. Clark, Stephan Lee, Dan Garrette, James Lee-Thorp, Colin Raffel, Noam Shazeer, Marvin Ritter, Maarten Bosma, Alexandre Passos, Jeremy Maitin-Shepard, Noah Fiedel, Mark Omernick, Brennan Saeta, Ryan Sepassi, Alexander Spiridonov, Joshua Newlan, and Andrea Gesmundo. 2022. Scaling up models and data with t5x and seqio. *arXiv preprint arXiv:2203.17189*.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. SIGIR '15, page 373–382, New York, NY, USA. Association for Computing Machinery.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved representation learning for question answer matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 464–473, Berlin, Germany. Association for Computational Linguistics.

MosaicML NLP Team. 2023. Introducing mpt-7b: A new standard for open-source, commercially usable llms. Accessed: 2023-05-05.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Gias Uddin and Foutse Khomh. 2017. Automatic summarization of api reviews. ASE '17, page 159–170. IEEE Press.

Di Wang and Eric Nyberg. 2015. A long short-term memory model for answer sentence selection in question answering. pages 707–712, Beijing, China. Association for Computational Linguistics.

Hanzhi Wang, Zhewei Wei, Junhao Gan, Sibo Wang, and Zengfeng Huang. 2020. Personalized pagerank to a target node, revisited. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 657–667, New York, NY, USA. Association for Computing Machinery.

Kai Wang, Zhaoyan Ming, and Tat-Seng Chua. 2009. A syntactic tree matching approach to finding similar questions in community-based qa services. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, page 187–194, New York, NY, USA. Association for Computing Machinery.

Mengqiu Wang and Christopher Manning. 2010. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1164–1172, Beijing, China. Coling 2010 Organizing Committee.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding.

Jiayuan Xie, Wenhao Fang, Qingbao Huang, Yi Cai, and Tao Wang. 2023. Enhancing paraphrase question generation with prior knowledge. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:1464–1475.

Bowen Xu, Zhenchang Xing, Xin Xia, and David Lo. 2017. Answerbot: Automated generation of answer summary to developers' technical questions. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 706–716.

Yumo Xu and Mirella Lapata. 2020. Coarse-to-fine query focused multi-document summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3632–3645, Online. Association for Computational Linguistics.

Zhaozhuo Xu, Zirui Liu, Beidi Chen, Yuxin Tang, Jue Wang, Kaixiong Zhou, Xia Hu, and Anshumali Shrivastava. 2023. Compress, then prompt: Improving accuracy-efficiency trade-off of llm inference with transferable prompt.

Chengran Yang, Bowen Xu, Jiakun Liu, and David Lo. 2023a. Techsumbot: A stack overflow answer summarization tool for technical query. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 132–135.

Chengran Yang, Bowen Xu, Ferdian Thung, Yucen Shi, Ting Zhang, Zhou Yang, Xin Zhou, Jieke Shi, Junda He, Donggyun Han, and David Lo. 2023b. Answer summarization for technical queries: Benchmark and new approach. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, ASE '22, New York, NY, USA. Association for Computing Machinery.

Zichun Yu, Chenyan Xiong, Shi Yu, and Zhiyuan Liu. 2023. Augmentation-adapted retriever improves generalization of language models as generic plug-in.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert.

Yufeng Zhang, Jinghao Zhang, Zeyu Cui, Shu Wu, and Liang Wang. 2021. A graph-based relevance matching model for ad-hoc retrieval.

Wenxuan Zhou, Sheng Zhang, Yu Gu, Muhao Chen, and Hoifung Poon. 2023. Universalner: Targeted distillation from large language models for open named entity recognition.

A   Appendix

A.1   Instruction tune hyperparameters

Hyperparameter	Value
Learning Rate	2e-4
Batch Size	4
Gradient Accumulation Step	1
Number of Epochs	10
Weight Decay	0.001
Optimizer	paged adamw 32bits
LR scheduler	cosine
Warmup ratio	0.03
Max grad norm	0.3
bf16	True
LoRA r, alpha, dropout	64, 16, 0.1
bf16	True
Quantization	4bit
PEFT Techniques	LoRA
Trainer	SFTT

Table 5: Hyperparameters for instruction tuning the LLM using SFTT trainer.

A.2   Pre-LLM era baselines

A.2.1   AnswerBot (Xu et al., 2017; Cai et al., 2019):

Authors of this work proposed an approach called AnswerBot, where the task is to generate a summary from diverse answers for a query. They followed three major steps – relevant question retrieval, useful answer paragraph selection, diverse answer summary generation. For retrieval, they used word2vec model and relevance calculation algorithm. For answer paragraph selection, they used various query, paragraph and user related features – *relevance to query*, *entity overlap*, *information*

entropy, *semantic pattern*, *format patterns*, *paragraph position*, *vote on answer*. In answer summary generation stage, they used maximal marginal relevance (MMR) algorithm to select a subset of answer paragraphs. Further they used selected answer paragraphs to form the answer summary.

A.2.2   GenQA (Hsu et al., 2021):

Authors of this paper proposed a framework to generate answers from the top candidates of a set of answer selection models. Instead of selecting the best candidates, they train a sequence to sequence transformer model to generate an answer from candidate set.

A.2.3   TechSumBot (Yang et al., 2023a):

Authors of this paper show that developers frequently turn to StackOverflow for solutions, but they often encounter redundant or incomplete results. Current tools designed to summarize StackOverflow answers have clear drawbacks: they predominantly depend on manually-designed features, they struggle to filter out repetitive content, and they usually target specific programming languages. This tool autonomously produces answer summaries by extracting and ranking answers for their relevance, measuring the core importance of each sentence, and eliminating redundant details. Presented in a search engine format, TechSumBot's efficiency is benchmarked against existing StackOverflow summary methods.

A.3   Sample prompt and generated answer

The sample prompt and the generated answer for a specific example is shown in Table 8.

A.4   Sample questions retrieved from our method

We include a few examples in Table 6 that take into account both PPR (graph structure-based) and simple similarity for certain questions. The questions retrieved by the PPR method are very specific, to-the-point, and strongly related to the actual question. The simple similarity-based questions, on the other hand, are very generic (e.g., What are some common network troubleshooting tools. . . , What are the best practices for managing log files. . .).

A.5   Enhanced context formulation

A.6   Human annotation

We engaged nine undergraduate students, each an expert in their domain, to undertake our annotation

Question	Related Question from PPR	Related Question from simple similarity
Experiencing Intermittent Network Failures on Ubuntu Server After Recent Update. **Desc:** After a recent update on my Ubuntu 20.04 server, I'm experiencing intermittent network failures. The server loses connectivity randomly, and I've been unable to diagnose the issue. Here's the output of `ifconfig` and `dmesg	grep eth0` after the failure occurs...	How can I rollback a recent Ubuntu update to troubleshoot network connectivity issues? **Desc:** Following recent network issues on my Ubuntu server, I suspect a recent update might be the cause. I need to rollback this update to confirm. What is the safest way to revert the last system update? Is there a way to identify which packages were updated and selectively rollback, or do I need to restore from a backup?	What are some common network troubleshooting tools in Ubuntu for diagnosing connectivity problems? **Desc:** In dealing with intermittent network failures on my Ubuntu server, I'm looking for effective tools or commands to diagnose the issue. What are the best tools available in Ubuntu for network troubleshooting, especially for monitoring and logging network activity over time to catch these intermittent failures?
Script for Automating Log File Rotation and Compression Not Working as Expected on Linux. **Desc:** I'm attempting to create a bash script to automate log file rotation and compression in a Linux environment. The script is supposed to find all log files under /var/log , compress them, and then move them to /archive/logs. However, it's not working as expected, and some log files are being missed...	How can I set up a cron job to run this script daily at midnight? **Desc:** I have a script for log file rotation and compression, but I'm not sure how to set it up as a cron job to run automatically. What is the correct way to schedule this script in cron to run daily at midnight? Are there any specific considerations for running such scripts as cron jobs?	What are the best practices for managing log files in a Unix environment? **Desc:** As I work on automating log file rotation and compression, I want to ensure I'm following best practices. What are the recommended strategies for log file management in a Unix environment? This includes considerations for log rotation frequency, compression, archiving, and ensuring log integrity and security?

Table 6: Retrieved questions from PPR (graph structure based) and simple similarity for certain questions.

ID	Error Type	Samples
1	Misaligned Retrieval Outcomes	Question: How can I change the desktop environment in Ubuntu? Retrieved Content: Steps to change the desktop wallpaper in Ubuntu. Generated Answer: To change the wallpaper , right-click on the desktop and select 'Change Wallpaper'........ Analysis: The retrieved content precisely discusses changing the wallpaper , but the user's query was about changing the entire desktop environment , not just the wallpaper.
2	Entity Misalignment	Question: What is the use of `chmod` in UNIX ? Retrieved Entity: `chown` command details from the KG . Generated Answer: .... `chown` is used to change the owner of a file or directory................................. Analysis: The retrieved information is accurate about `chown` , but the user's query was about `chmod` . This represents a wrong entity linking.
3	Composite Query Conundrums	Question: How do I configure the firewall on my server to allow SSH access , and also set up an FTP server for file transfers? Retrieved Content: Detailed steps on configuring the firewall for SSH access . Generated Answer: To configure the firewall for SSH access , you need to open port 22 and apply the necessary rules... Analysis: While the retrieved content and generated answer correctly address the firewall configuration for SSH , the question's second aspect about setting up an FTP server is entirely neglected.
4	Factual Fidelity Fallacies	Question: How can I upgrade to the latest version of Ubuntu ? Retrieved Content: Steps to upgrade Ubuntu 16.04 to Ubuntu 18.04 . Generated Answer: To upgrade to the latest version of Ubuntu , follow these steps to upgrade from Ubuntu 16.04 to 18.04 ... Analysis: While the retrieved and generated content might be factually correct for the specific version transition, it's outdated if the latest version is beyond Ubuntu 18.04 .
5	Contextual Content Crux	Question: How do I solve the XYZ error in Ubuntu? Retrieved Content: A post briefly mentioning the XYZ error with a comment saying, "Ï also faced this." Generated Answer: The XYZ error is an issue some users have faced in Ubuntu... Analysis: The retrieved content acknowledges the error but provides no solution or detailed information, leading to an unsatisfactory and unhelpful answer.

Table 7: The actual rationale being marked with the green and retrieved and generated rationale marked as red .

Figure 4: Sample context preparation.

task, dividing them into groups of three for each dataset. All these students are majoring in Computer Science and have a proven track record of contributing high-quality answers within relevant community platforms. They voluntarily joined our project after receiving an invitation through departmental emails and were rewarded with Amazon gift cards for their contributions. Each undergraduate student annotates 50 instances individually. We utilize the Doccano[15] tool for obtaining the annotations. The annotators provided feedback on a scale from 1 to 5. A rating of '1' means the answer is unhelpful or misleading, while a '5' indicates an exemplary response. In Figure 3, we display the

---
[15]https://elements.heroku.com/buttons/doccano/doccano

Question	How do I extract an ISO file?
Context	Question: Has anyone used isoInfo to extract ISO files? How does it compare to others? Answer: Hey! So, 'isoInfo' isn't for extracting ISO files; it just shows details about them. You wanna extract? Use 'isoinfo' from the same package. Like this: isoinfo -i image.iso -x /PATH/INSIDE/ISO >output.file Now, comparing 'isoinfo' with other tools: 1. 7-Zip: Cool for many file types, has both GUI and command-line. 2. WinRAR: Good for ISOs, but it's paid (though many stick to the trial). 3. Daemon Tools: Great for 'mounting' ISOs, but can extract too. 4. dd & mount (Linux stuff): More techie, but pretty powerful. If you're a Linux person comfy with commands, go for 'isoinfo'. If not, maybe try a tool with a friendly interface. Question: Can you recommend a good free tool for extracting ISO files on Ubuntu 20.04? Answer: Hey! On Ubuntu 20.04 and wanna extract ISO files? Easy peasy: 1. Archive Manager: Built-in tool. Just right-click the ISO, hit "Open With Archive Manager", and boom, you're in. 2. 7-Zip: Not just a Windows thing! Get the Linux version, 'p7zip', with: sudo apt install p7zip-full Then run: 7z x file.iso 3. mount: A bit techie. Mount the ISO, copy its stuff, like: mkdir /path/to/mountpoint sudo mount -o loop file.iso /path/to/mountpoint cp -r /path/to/mountpoint /path/to/destination sudo umount /path/to/mountpoint 4. isoinfo: Fancy extracting specific stuff? Grab it with: sudo apt install genisoimage Most folks will find Archive Manager or 7-Zip super easy. But if you're a command line junkie, try 'mount' or 'isoinfo'. Question: What could be the important context to answer this? Answer: ('Archive Manager is a built-in tool'), ('Archive Manager is used for extracting ISO files'), ('7-Zip is a Linux version'), ('7-Zip is used to extract ISO files'), ('7-Zip is used for extracting ISO files'), ('WinRAR is a paid tool'), ('Daemon Tools is used for mounting ISOs'), ('dd & mount is a Linux stuff'), ('isoinfo is used for extracting specific stuff'), ('isoinfo is used for extracting ISO files'), ('mount is used for extracting ISO files')
Answer	To extract an ISO in Linux/Ubuntu: 1. Archive Manager: Built-in. Right-click the ISO, select "Open With Archive Manager", and extract. 2. 7-Zip: Install using 'sudo apt install p7zip-full' and extract with '7z x file.iso'. 3. mount: Create a mount point 'mkdir /mount/spot'. Mount with 'sudo mount -o loop file.iso /mount/spot', copy with 'cp -r /mount/spot /destination', then unmount 'sudo umount /mount/spot'. 4. isoinfo: Install with 'sudo apt install genisoimage'. To extract specifics, use 'isoinfo -i image.iso -x /PATH/INSIDE/ISO >output.file'. For ease, go with Archive Manager or 7-Zip. If you're command-line savvy, try 'mount' or 'isoinfo'.

Table 8: Sample prompt an generated answer.

feedback distribution for test instances. In the pie chart, the outer ring represents the three datasets. For every dataset, five segments in the inner ring depict the distribution of ratings from 1 to 5. The plot reveals that the answers generated by our model for both the AskUbuntu and the Unix test cases predominantly have a rating of 4 as per human judgement, while those generated by our model for the ServerFault test cases predominantly have ratings of 5. Next we compute 'win rate' which refers to the percentage of individuals who favor the output from our model over the standard zero-shot output. In our analysis comparing answers generated by our model with those from a simple zero-shot approach, we observe a notable trend in win rates across the three platforms. Specifically, for Askubuntu, Unix, and ServerFault, the win rates are 58%, 63%, and 53%, respectively. These rates consistently exceed the 50% benchmark.