REGULAR: A Framework for Relation-Guided Multi-Span Question Generation

Anonymous ACL submission

Abstract

To alleviate the high cost of manually annotating Question Answering (QA) datasets, Question Generation (QG) has been proposed, which requires the model to generate a question related to the given answer and passage. This work primarily focuses on Multi-Span Question Generation (MSQG), where the generated question corresponds to multiple candidate answers. We observe that traditional QG methods may not suit MSQG as they typically overlook the correlation between the candidate answers and generate trivial questions. To address it, we propose **REGULAR**, a framework of **RE**lation-**GU**ided Mu**L**ti-Sp**A**n Question Gene**R**ation. REGULAR first converts passages into knowledge graphs and extracts candidate answers from the knowledge graphs. Then, REGULAR utilizes a QG model to generate a set of candidate questions and a QA model to obtain the optimal question. We construct over 100,000 questions using Wikipedia and PubMed corpora, named REGULAR-WIKI and REGULAR-MED respectively, and conduct experiments to compare our synthetic datasets with other synthetic QA datasets. The experiment results show that models pre-fine-tuned with our synthetic dataset achieve optimal performance. We also conduct ablation studies and statistical analysis to verify the quality of our synthetic dataset. [1]

1 Introduction

Question Answering (QA) (Rajpurkar et al., 2018; Kwiatkowski et al., 2019) requires the model to provide accurate answers for a given question, which has wide-ranging applications like chat systems(OpenAI et al., 2024), information retrieval(Esteva et al., 2021), and AI education (Rabin et al., 2023). As a subtype of the QA task, Multi-Span Question Answering (MSQA) (Li et al., 2022; Yue et al., 2023) requires the model to extract multiple non-redundant answers from a given passage. However, the models may need a large amount of training data to facilitate either MSQA or other QA tasks. To alleviate the high cost of manually annotating QA datasets, Question Generation (QG) has been proposed, which requires the model to generate a question related to the given answer and passage.

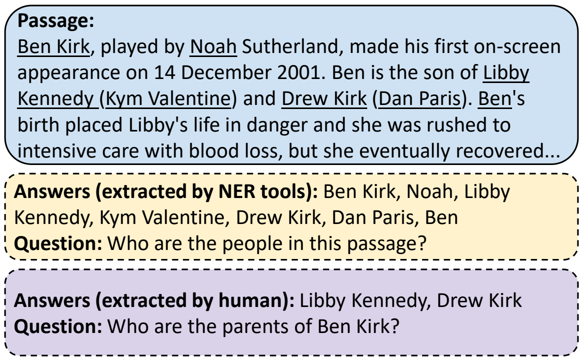

Figure 1: An example where humans and the NER tool extracted different answers, leading to different questions. The entities extracted by the NER tool are highlighted with underlines.

Traditional QG research (Shakeri et al., 2020; Lyu et al., 2021a; Lee et al., 2023) has considered cases where the answer is not provided, meaning that the task is to generate a question and its corresponding answer from a given passage. Existing methods typically use rule-based methods or model-based methods to extract candidate answers, and then employ a Language Model (LM) to generate the question. For example, Shakeri et al. (2020) use a Sequence-to-Sequence LM (Sutskever et al., 2014) to generate questions and answers in an end-to-end manner, while Lyu et al. (2021a) and Lee et al. (2023) utilize Named Entity Recognization (NER) tools to extract answers. Some other works (Guo et al., 2024; Wu et al., 2024) explore improving the ability of Large Language Models (LLMs)

[1] Our code and data are available at `https://anonymous.4open.science/r/REGULAR-BC26`

to generate questions by incorporating additional information into the prompt.

This work primarily focuses on Multi-Span Question Generation (MSQG), where the generated question corresponds to multiple candidate answers. Unfortunately, traditional QG struggles with MSQG. Taking Figure 1 as an example, using NER tools to extract all the person names results in a set of unrelated entities, which leads to a trivial question. The reason may be that traditional QG methods primarily focus on generating single-answer questions, without considering the correlation between multiple answers in the MSQG task. Although LIQUID (Lee et al., 2023) employs an additional QA model to refine the initial candidate answers, the correlation between candidate answers is still not guaranteed.

We observe that knowledge graphs may help obtain relevant candidate answers, as edges connect different entities with relationship types in the knowledge graph. Based on this observation, we define Commonality Entities (CE) as a group of entities that share the same relation type with a specific entity in a knowledge graph. Then we propose **REGULAR**, a framework for **RE**lation-**GU**ided Mu**L**ti-Sp**A**n Question Gene**R**ation. For a given passage, REGULAR converts it into a knowledge graph and employs a graph traversal algorithm to extract CE as candidate answers. After extracting candidate answers, REGULAR utilizes a QG model to generate a set of candidate questions and a QA model to obtain the optimal question. Compared with traditional QG methods, REGULAR considers the relevance between candidate answers, avoiding the negative impact of irrelevant answers on the synthetic datasets.

We construct over 100,000 questions using Wikipedia and PubMed corpora, named REGULAR-WIKI and REGULAR-MED respectively[2], and evaluate them through 2-step fine-tuning experiments. The experiment results show that models pre-fine-tuned with REGULAR dataset achieve optimal performance. For instance, after training with REGULAR-WIKI, the Tagger model (Li et al., 2022) improves the Exact Match F1 by 2.95% on MultiSpanQA(Li et al., 2022) compared with the model pre-fine-tuned with the LIQUID dataset (Lee et al., 2023). Additionally, we conduct ablation studies and statistical analysis to verify the quality of the REGULAR datasets.

[2]For simplicity, we also refer them to REGULAR datasets.

In summary, our contributions are listed as follows:

- To obtain relevant candidate answers in MSQG, we explore extracting entities from the knowledge graph as candidate answers. We define CE as a group of entities that share the same relation type with a specific entity in a knowledge graph and design a graph traversal algorithm to extract CE.
- We propose REGULAR, which extracts CE from graph structures as candidate answers and generates corresponding questions. We construct over 100,000 questions from Wikipedia and PubMed corpora, respectively.
- Experiment results demonstrate that our synthetic datasets can be used to train QA models and achieve better performance. We also conduct ablation studies and statistical analysis to validate the quality of the synthetic dataset.

2 Related Work

2.1 Question Generation

QG requires models to generate a question that matches the given passage and the answer. This work primarily focuses on MSQG where the generated question corresponds to multiple answers. In real-world applications, the answers are often unknown, so obtaining the answers is necessary first and then generating the corresponding questions.

Traditional methods typically utilize LMs or rule-based tools to extract candidate answers. Puri et al. (2020) train a BERT (Devlin et al., 2019) to extract candidate answers. Shakeri et al. (2020) use a Sequence-to-Sequence LM to end-to-end generate both questions and answers. Lyu et al. (2021a) extract summarization of the given passage and then use NER tools and syntactic parsing tools to extract candidate answers. LIQUID (Lee et al., 2023) first extracts multiple candidate answers with a summarization model and NER tool, and generates multi-answer questions, followed by iterative updates to both the questions and candidate answers. However, these methods fail to consider the correlation between candidate answers. In contrast, we extract CE in the knowledge graph, ensuring the correlation among the candidate answers and improving the quality of the synthetic datasets.

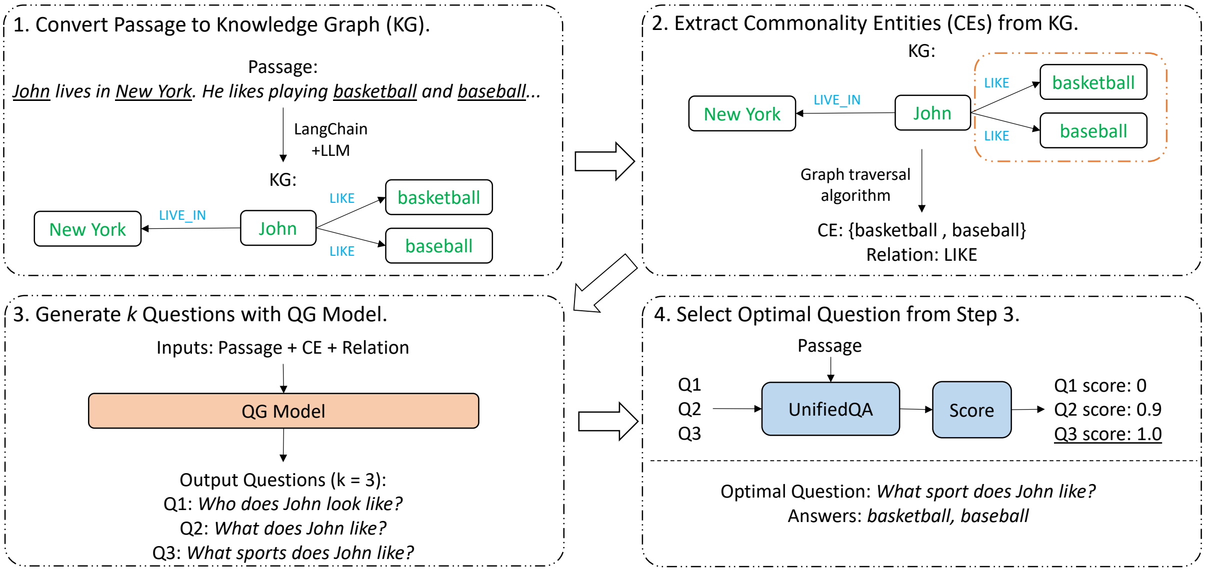

Figure 2: The pipeline of our REGULAR framework.

2.2 LLM-based Question Generation

Recently, LLMs (Grattafiori et al., 2024; OpenAI et al., 2024) have gained widespread attention due to their powerful language modeling and text generation capabilities. Recent studies have explored methods such as In-Context Learning (ICL) (Brown et al., 2020) and Chain-of-Thought (CoT) (Wei et al., 2022; Kojima et al., 2022) to further improve the performance of LLMs in QG tasks.

For example, TASE-CoT (Lin et al., 2024) first uses the T5 (Raffel et al., 2020) model to predict the question type and key fragments within the question, then designs a three-step CoT approach to guide the LLM in generating multi-hop questions. Similarly, SGSH (Guo et al., 2024) addresses Knowledge Base Question Generation (KBQG) by using a fine-tuned BART (Lewis et al., 2020) model to provide the question prefix before generating questions with GPT-3.5. Li and Zhang (2024) focus on controllable question generation and propose the PFQS framework. This framework first generates an initial plan based on the question label, adjusts it with the context, and then generates the question based on the article, answer, and plan. In addition to text-only question generation, Wu et al. (2024) focus on Multi-Modal Question Generation (MMQG) and they propose SMMQG, which samples multi-modal sources and generates different types of questions with GPT-4.

In this work, we primarily utilize advanced LLMs to convert passages into knowledge graphs and use fine-tuned LLMs to generate questions.

3 Method

The MSQG task can be described as: Given a passage p, models are required to first extract a set of non-redundant text spans as the candidate answers A, and then generate the corresponding question q, as shown in Equation 1:

$$
\begin{aligned}
A &= Extract\_Answers(p) \\
q &= M_{QG}(p, A)
\end{aligned} \tag{1}
$$

where M_{QG} refers to the QG model. Figure 2 shows the architecture of our REGULAR framework. Different from existing work, we extract CE from the knowledge graphs as the candidate answers to ensure relevance among the answers and then generate corresponding questions. Specifically, the REGULAR framework consists of four steps: (1) Convert the given passage to a knowledge graph; (2) Extract CE from the knowledge graph as candidate answers; (3) Utilize a QG model to generate a set of candidate questions; (4) Score each candidate question with a QA model and select the optimal question with the highest score for constructing the MSQA dataset.

Next, we will introduce the definition of CE in Section 3.1, and elaborate on each step from Section 3.2 to Section 3.4.

3.1 Commonality Entities

The definition of CE can be described as follows: Given a reference entity $\bar{v}$ and a relation r, CE is defined as a set of entities that connect to $\bar{v}$ with the edges that share the same relation r. The above definition can be represented by Equation 2.

$$CE(\bar{v}, r) = \{v | v \in N(\bar{v}) \wedge R(v, \bar{v}) = r\} \quad (2)$$

where $N(\bar{v})$ represents the neighbor entities of $\bar{v}$ and $R(v_i, \bar{v})$ represents the relation of the edge between v_i and $\bar{v}$.

3.2 Extracting CE as candidate answers

In MSQG tasks, selecting multiple candidate answers is important because unrelated candidate answers may result in low-quality questions (Lyu et al., 2021b; Lee et al., 2023). Existing methods (Lee et al., 2023) typically utilize NER tools (e.g., SpaCy [3]) to extract named entities. However, these approaches fail to consider the correlations among candidate answers, thereby limiting the quality of the synthetic data.

We propose extracting CE as candidate answers, considering that CE in a knowledge graph is connected to a specific entity through the same edges, ensuring relevance among these entities. This process contains two steps: converting passages into knowledge graphs and extracting CE in the knowledge graph.

Converting Passages into Knowledge Graphs We utilize *LangChain LLMGraphTransformer*[4] to convert passages into knowledge graphs. This process can be described as Equation 3:

$$G = LLM(p) \quad (3)$$

where p refers to the passage and G refers to the knowledge graph and $LLM()$ refers to the LangChain tool.

Extracting CE in the Knowledge Graph We design a graph traversal algorithm that identifies CE by counting the 1-hop neighbors of each node. We extract CE with two or more entities as candidate answers A. This process can be described as Equation 4:

$$A = Extract\_Answers(G) \quad (4)$$

where G refers to the knowledge graph. We provide a detailed algorithm in Appendix A.

[3] https://spacy.io/

[4] https://python.langchain.com/api_reference/experimental/graph_transformers/langchain_experimental.graph_transformers.llm.LLMGraphTransformer.html

3.3 Generating Questions

Generating Questions with CE We utilize a generative LM M_{QG} as the QG model to generate questions. The inputs of M_{QG} are the passage p, the candidate answers A, reference entity v, and relation r. We sample k candidate questions $Q = \{q_1, ..., q_k\}$, where k is the number of generated questions, shown in Equation 5:

$$Q = M_{QG}(p, A, \hat{v}, r) \quad (5)$$

Extracting Relations for Training the QG Model Existing MSQA datasets such as MultiSpanQA(Li et al., 2022) and MA-MRC(Yue et al., 2023) do not include commonality relation we need. Intuitively, we could use a prompted LLM to extract the commonality relation from the question. However, this may introduce bias between training and generating. To address this problem, we first prompt an LLM to convert the question-answer pairs into declarative sentences. Then, following the method proposed in Section 3.2, we check whether the answers satisfy the definition of CE. If the candidate answers are CE, we add the corresponding commonality relation r to the training data, otherwise, we discard this data.

3.4 Obtaining Optimal Question

Existing QG researches (Lee et al., 2023; Mohammadshahi et al., 2023) typically employ a QA model to validate the generated questions. In this work, we employ a QA model M_{QA} fine-tuned on the MSQA datasets to score the candidate questions generated in Section 3.3 and select the question with the highest score. For each candidate question $q_i \in Q$ and its corresponding passage p, we predict its answers with M_{QA}. Then we calculate the F1 score of the predicted answers and obtain the optimal question $\hat{q}$ that maximizes the F1 score. This process can be described as Equation 6:

$$\begin{aligned} O_i &= M_{QA}(p, q_i) \\ s_{q_i} &= F1\_Score(O_i, A) \\ \hat{q} &= \underset{q_i \in Q}{argmax}(s_{q_i}) \end{aligned} \quad (6)$$

where $F1\_Score(O_i, A)$ refers to the F1 score of O_i when A is used as the reference[5].

Finally, we construct synthetic dataset D with the candidate answers A and the generated question

[5] When calculating the F1 score, we take the average of the Exact Match F1 and Partial Match F1 scores. Details of Exact Match and Partial Match are shown in Section 4.1

	MultiSpanQA		MA-MRC		QUOREF	
	EM F1	PM F1	EM F1	PM F1	EM F1	PM F1
Tagger	68.58	83.62	79.69	87.50	66.77	81.78
+ QAGen-WIKI	67.04	82.89	79.48	87.19	68.86	82.20
+ LIQUID-WIKI	67.50	83.26	**80.31**	87.59	73.91	**85.67**
+ REGULAR-WIKI	**70.45**	**84.57**	80.27	**87.99**	**75.89**	84.77
SpanQualifier	71.58	83.50	79.85	86.83	62.21	75.36
+ QAGen-WIKI	69.81	83.26	57.68	73.74	58.29	73.02
+ LIQUID-WIKI	70.76	**84.52**	80.66	87.28	71.95	80.31
+ REGULAR-WIKI	**72.98**	83.19	**81.38**	**88.06**	**72.98**	**83.19**
BART-base	65.14	80.15	75.38	84.40	66.80	75.38
+ QAGen-WIKI	66.98	81.32	74.29	83.97	61.97	73.62
+ LIQUID-WIKI	66.67	81.14	75.84	84.60	67.24	77.17
+ REGULAR-WIKI	**68.98**	**82.59**	**75.88**	**85.28**	**67.44**	**78.51**
T5-base	69.24	82.93	78.39	86.21	65.47	76.40
+ QAGen-WIKI	69.81	83.26	78.89	86.32	61.41	73.25
+ LIQUID-WIKI	70.29	81.83	79.26	86.78	67.64	75.94
+ REGULAR-WIKI	**73.09**	**85.19**	**79.40**	**86.87**	**67.66**	**76.23**

Table 1: Exact Match and Partial Match F1 scores of the MSQA models. The first line of each MSQA model refers to the original performance. "QAGen-WIKI" and "LIQUID-WIKI" refer to the 2-step fine-tuning baselines where models are firstly trained on synthetic datasets from Wikipedia corpus. The best results are in **bold**.

$\hat{q}$, shown in Equation 7:

$$D = \{(p_j, A_j, \hat{q}_j)\}_{j=1}^{n} \tag{7}$$

where n refers to the question number of D.

4 Experiments

Inspired by (Lee et al., 2023), we use a 2-step fine-tuning approach to compare the quality differences of the synthetic datasets generated by REGULAR and other QG methods. In the first step, the QA models are pre-fine-tuned on the synthetic datasets, and in the second step, the QA models are grained-fine-tuned on the downstream MSQA benchmarks. For a fair comparison, we randomly select 50,000 questions for the pre-fine-tuning.

4.1 Experimental Setup

Corpus We select the open-source corpus **PubMed** [6] and **Wikipedia** [7] and construct over 100,000 questions using each of these two corpus, named REGULAR-WIKI and REGULAR-MED respectively. The PubMed corpus focuses on the biomedical field, while Wikipedia covers general knowledge.

QG Baselines We select synthetic datasets, including **QAGen** (Shakeri et al., 2020), **LIQUID-MED** (Lee et al., 2023) and **LIQUID-WIKI** (Lee et al., 2023) as the baselines. Details of these baselines are shown in Appendix B.3.

[6] https://pubmed.ncbi.nlm.nih.gov/

[7] https://www.wikipedia.org/

MSQA Datasets We select the **MultiSpanQA** (Li et al., 2022), **MA-MRC** (Yue et al., 2023), and **QUOREF** (Dasigi et al., 2019) for our experiments. Considering that the MA-MRC dataset contains a large amount of training data, we randomly sample 10,000 training data and 1,000 validation data and obtain **MA-MRC-10k**. Details of the MSQA dataset are shown in Appendix B.1.

MSQA Models We select two discriminative models: **Tagger** (Li et al., 2022) and **SpanQualifier** (Huang et al., 2023),as well as two generative models: **BART** (Lewis et al., 2020) and **T5** (Raffel et al., 2020) for our experiments. Details of these models are shown in Appendix B.2.

Evaluation Metrics Following (Li et al., 2022), we use **Exact Match (EM)** and **Partial Match (PM)** as the main metrics. EM assigns a score of 1 when a prediction fully matches one of the gold answers and 0 otherwise, while PM considers the overlap between the predictions and gold answers. We report F1 scores in our experiments.

Implementation Details Implementation details are shown in Appendix B.4.

4.2 Main Results

The main results are shown in Table 1 and Appendix Table 5. Based on these results, the following conclusions can be made: **(1) Traditional QG Methods (e.g., QAGen (Shakeri et al., 2020)) are not suitable for constructing MSQA datasets.** We observe that after pre-fine-tuning with the QAGen dataset, the model's performance decreases in

	MultiSpanQA	
	EM F1	**PM F1**
Tagger	68.58	83.62
Ablation on Question Generation Steps		
w/o KG (Step 1)	67.52	83.16
w/o CE (Step 2)	68.06	71.53
w/o relation (Step 3)	70.48	84.61
Random Question (Step 4)	67.44	83.16
Worst Question (Step 4)	66.65	81.98
REGULAR	**70.51**	**85.14**
Ablation on Fine-Tuning Strategies		
Merged FT	70.03	84.98
Domain-Shift FT	60.25	75.51
REGULAR	**70.51**	**85.14**

Table 2: Ablation Study on question generation steps and fine-tuning strategies on the validation set of MultiSpanQA. The best results are in **bold**.

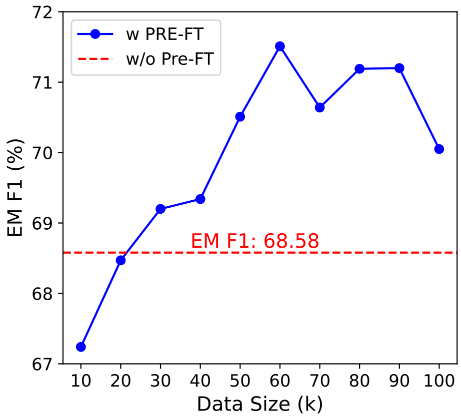

Figure 3: Ablation study on data scale with Tagger on the validation set of MultiSpanQA. "w PRE-FT" refers to the pre-fine-tuning results and "w/o PRE-FT" refers to the original result without pre-fine-tuning.

most settings. This suggests that synthetic datasets generated by QAGen do not contribute to improving the performance of the MSQA models; **(2) The LIQUID datasets slightly improve the performance of MSQA models in some settings.** For instance, on the MA-MRC-10k dataset, the LIQUID-MED setting improved the EM F1 score of the Tagger model from 79.69 to 80.31. However, in other settings, the performance shows a slight decline. This indicates that the quality of the LIQUID dataset still needs improvement. **(3) Our synthetic datasets perform best in most settings.** This is because the REGULAR framework extracts CE from the knowledge graph, ensuring the correlation between candidate answers, and thereby improves the quality of the synthetic dataset.

We also conduct experiments with the open-source LLMs. Details and results are shown in Appendix C.2.

4.3 Ablation Study

Ablation on Question Generation Steps We hypothesize that each step in REGULAR contributes to constructing a higher-quality synthetic dataset. To validate this, we conduct ablation studies on each synthetic step of REGULAR and evaluate the validation set of the MultiSpanQA dataset. We implement the following ablation strategies: (1) **w/o KG**: Use NER tools to extract candidate entities from the passage. (2) **w/o CE**: Randomly select entities and their neighbors as candidate answers instead of CE. (3) **w/o relation**: Remove the commonality relation and key entity when generating questions. (4) **Random Question**: Randomly select a candidate question instead of the highest-scoring question. (5) **Worst Question**: Select the lowest-scoring question instead of the highest-scoring question.

As shown in Table 2, all ablation settings lead to a decline in model performance. Notably, the ablation of Step 1 and Step 2 results in a significant drop in performance, as the candidate answers selected under these conditions lack correlation, which limits the quality of the synthetic dataset. Furthermore, randomly selecting questions or choosing the worst-performing questions also has a negative impact, indicating that the quality of the generated questions also influences the overall quality of the synthetic dataset.

Ablation on Fine-Tuning Strategies Inspired by Lee et al. (2023), we employ a 2-step fine-tuning strategy to demonstrate the quality of the synthetic dataset. To explore whether other fine-tuning strategies might perform better, we test two other fine-tuning strategies: (1) **Merge FT**: We mix the synthetic dataset with the downstream benchmark dataset and fine-tune them simultaneously. (2) **Domain-Shift FT**: We only fine-tune models on the synthetic dataset.

We compare these strategies on the MultiSpanQA validation set. As shown in Table 2, Merge FT and Domain-Shift FT perform worse than the 2-step fine-tuning strategy. We hypothesize that the model learns to reason, generalize, and summarize within context using a large amount of data in the pre-fine-tuning phase. In contrast, in the grained-fine-tuning phase, the model adapts to the domain and question format of the downstream tasks, which leads to better performance on the validation set.

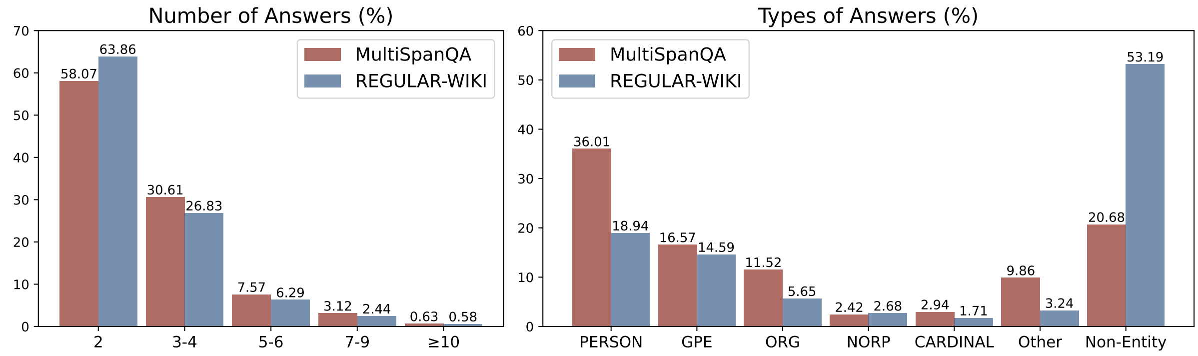

Figure 4: Left: Number of answers in the MultiSpanQA and the REGULAR-WIKI datasets; Right: Types of answers in the MultiSpanQA and the REGULAR-WIKI datasets. The numbers in the figures represent the percentage.

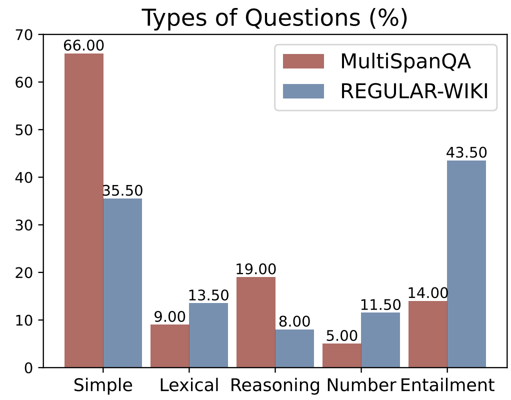

Figure 5: Types of questions in the MultiSpanQA and the REGULAR-WIKI datasets.

Ablation on Data Scale To investigate the impact of dataset scale on the fine-tuning results, we conduct pre-fine-tuning on Tagger with dataset sizes ranging from 10,000 to 100,000, followed by grained-fine-tuning on the MultiSpanQA dataset. As shown in Figure 3, the model achieves optimal performance when the pre-fine-tuning dataset size reaches 60,000. When the pre-fine-tuning dataset size is too small, the model performs worse than the original result. This may be because a smaller dataset cannot fully leverage the advantages of pre-fine-tuning. On the other hand, when the dataset size exceeds a certain threshold, the model's performance does not improve further. This suggests that choosing an appropriate dataset size for pre-fine-tuning is important.

5 Analysis on the Synthetic Dataset

In this section, we statistically analyze the answer types, number of answers, and question types in the REGULAR-WIKI and MultiSpanQA datasets. We also conduct a case study to compare REGULAR-WIKI with QAGen-WIKI. The analysis for the REGULAR-MED dataset is presented in Appendix E.

5.1 Number of Answers

We analyze the number of answers for each question in the MultiSpanQA and REGULAR-WIKI datasets, as shown in Figure 4. Compared with the MultiSpanQA dataset, the REGULAR-WIKI dataset has a higher proportion of questions with 2 answers and a lower proportion with more than 3 answers. This may be because REGULAR extracts answers with specific topological structures (i.e. CE), limiting the number of answers.

5.2 Types of Answers

We use SpaCy to analyze the answer types in the MultiSpanQA and REGULAR-WIKI datasets. Figure 3 shows the proportion of named entity answers with top-5 frequencies and non-named entity answers. Surprisingly, we observe that the proportion of non-entity answers in REGULAR-WIKI was much higher than in MultiSpanQA. This may be because both named and non-named entities were included as nodes during the knowledge graph extraction process. The reason may be that incorporating more non-named entities as candidate answers helps enhance the diversity of questions and answers.

5.3 Types of Questions

We further analyze the distribution of question types in REGULAR-WIKI and MultiSpanQA datasets. We adopt the categories proposed by Lee et al. (2023): Simple Questions, Lexical Variation, Inter-sentence Reasoning, Number of Answers, and Entailment, where a question may correspond to multiple types. We sample 200 questions and use

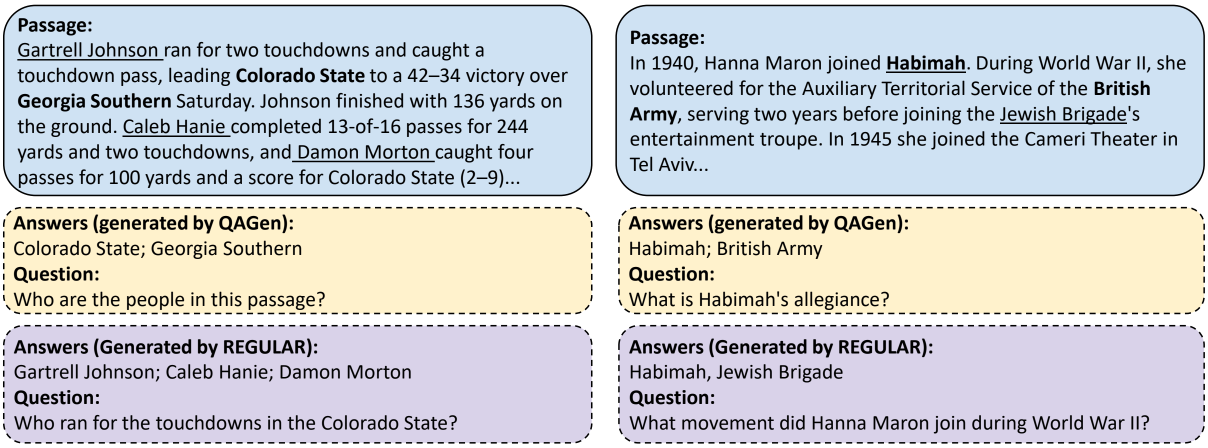

Figure 6: Case study. The examples are selected from the QAGen-WIKI and REGULAR-WIKI. We mark answers of QAGen-WIKI with **bold** and REGULAR-WIKI with underline. The numbers in the figure represent the percentage.

GPT-4o to classify each question. Detailed definitions of the five types of questions can be found in Appendix D.

The statistical results are shown in Figure 5[8]. Compared with the MultiSpanQA dataset, the REGULAR-WIKI dataset contains fewer Simple Questions. These questions typically have answers within a single sentence, but the answers in REGULAR-WIKI are derived from knowledge graphs and might span multiple sentences. On the other hand, REGULAR-WIKI contains more Entailment questions, perhaps because the generated questions implicitly contain prior knowledge from the QG model. Overall, the question distribution in REGULAR-WIKI is more balanced, suggesting that the REGULAR framework can generate a wider variety of questions.

5.4 Case Study

We conduct a case study demonstrating that the REGULAR method can generate better synthetic datasets. Figure 6 shows examples of questions and answers generated by QAGen and REGULAR for the same passage. In the first example, QAGen generates an inaccurate question, "Who is the host of Gartell Johnson?" Although the question is grammatically correct, the corresponding answers, "Colorado State" and "Georgia Southern" do not match the question. In contrast, REGULAR extracts three names from the knowledge graph, all of whom participate in the game, and thus the question, "Who ran for the touchdowns in Colorado State?" is more accurate. Similarly, in the second example, REGULAR extracts two organizations Hanna Maron joined during World War II and generates the corresponding question. These examples demonstrate that the REGULAR method, by extracting CE, can generate higher-quality questions and answers.

[8]Due to differences in sampling data and evaluation methods, the analysis results may differ from the results in (Lee et al., 2023).

6 Conclusion

In this work, we focus on the MSQG task and propose REGULAR, a framework of relation-guided Multi-Span Question Generation. REGULAR converts passages into knowledge graphs and extracts CE as the candidate answers. Then, REGULAR utilizes a QG model to generate a set of candidate questions and a QA model to obtain the optimal question. We construct over 100,000 questions using Wikipedia and PubMed corpora, named REGULAR-WIKI and REGULAR-MED respectively, and conduct 2-step fine-tuning experiments. The experiment results show that models pre-fine-tuned with the REGULAR dataset achieve optimal performance, indicating that the quality of the REGULAR datasets is higher than other synthetic QA datasets.

7 Limitations and Future Work

In this work, we utilize LangChain to convert passages into knowledge graphs. However, this step relies on advanced LLMs (e.g., GPT-4o-mini), which may incur significant costs. Although we assume that advanced LLMs have mastered the ability to extract knowledge graphs during their training, we have not explicitly addressed the potential errors that may occur. On the other hand, we primarily focus on generating multi-answer questions. We do not consider other types of question generation (e.g., multi-hop reasoning questions, multiple-choice questions, etc.).

In future work, we plan to improve the ability of LLMs to extract knowledge graphs with the open-source LLMs (e.g., Llama, Qwen). Additionally, we will explore how this method can be applied to generate other types of questions.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Pradeep Dasigi, Nelson F. Liu, Ana Marasović, Noah A. Smith, and Matt Gardner. 2019. Quoref: A reading comprehension dataset with questions requiring coreferential reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5925–5932, Hong Kong, China. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Andre Esteva, Anuprit Kale, Romain Paulus, Kazuma Hashimoto, Wenpeng Yin, Dragomir Radev, and Richard Socher. 2021. Covid-19 information retrieval with deep-learning based semantic search, question answering, and abstractive summarization. *NPJ digital medicine*, 4(1):68.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Shasha Guo, Lizi Liao, Jing Zhang, Yanling Wang, Cuiping Li, and Hong Chen. 2024. SGSH: Stimulate large language models with skeleton heuristics for knowledge base question generation. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 4613–4625, Mexico City, Mexico. Association for Computational Linguistics.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Zixian Huang, Jiaying Zhou, Chenxu Niu, and Gong Cheng. 2023. Spans, not tokens: A span-centric model for multi-span reading comprehension. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, CIKM '23, page 874–884, New York, NY, USA. Association for Computing Machinery.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*.

Seongyun Lee, Hyunjae Kim, and Jaewoo Kang. 2023. Liquid: A framework for list question answering dataset generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(11):13014–13024.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL:2020:main*, pages 7871–7880, Online. acl.

Haonan Li, Martin Tomko, Maria Vasardani, and Timothy Baldwin. 2022. MultiSpanQA: A dataset for multi-span question answering. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1250–1260, Seattle, United States. Association for Computational Linguistics.

Kunze Li and Yu Zhang. 2024. Planning first, question second: An LLM-guided method for controllable question generation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 4715–4729, Bangkok, Thailand. Association for Computational Linguistics.

Zefeng Lin, Weidong Chen, Yan Song, and Yongdong Zhang. 2024. Prompting few-shot multi-hop question generation via comprehending type-aware semantics. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3730–3740, Mexico City, Mexico. Association for Computational Linguistics.

Chenyang Lyu, Lifeng Shang, Yvette Graham, Jennifer Foster, Xin Jiang, and Qun Liu. 2021a. Improving unsupervised question answering via summarization-informed question generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4134–4148, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Chenyang Lyu, Lifeng Shang, Yvette Graham, Jennifer Foster, Xin Jiang, and Qun Liu. 2021b. Improving unsupervised question answering via summarization-informed question generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4134–4148, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Alireza Mohammadshahi, Thomas Scialom, Majid Yazdani, Pouya Yanki, Angela Fan, James Henderson, and Marzieh Saeidi. 2023. RQUGE: Reference-free metric for evaluating question generation by answering the question. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6845–6867, Toronto, Canada. Association for Computational Linguistics.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Raul Puri, Ryan Spring, Mohammad Shoeybi, Mostofa Patwary, and Bryan Catanzaro. 2020. Training question answering models from synthetic data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5811–5826, Online. Association for Computational Linguistics.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Roni Rabin, Alexandre Djerbetian, Roee Engelberg, Lidan Hackmon, Gal Elidan, Reut Tsarfaty, and Amir Globerson. 2023. Covering uncommon ground: Gap-focused question generation for answer assessment. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 215–227, Toronto, Canada. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Siamak Shakeri, Cicero Nogueira dos Santos, Henghui Zhu, Patrick Ng, Feng Nan, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. 2020. End-to-end synthetic data generation for domain adaptation of question answering systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5445–5460, Online. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3104–3112, Cambridge, MA, USA. MIT Press.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Ian Wu, Sravan Jayanthi, Vijay Viswanathan, Simon Rosenberg, Sina Khoshfetrat Pakazad, Tongshuang Wu, and Graham Neubig. 2024. Synthetic multimodal question generation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 12960–12993, Miami, Florida, USA. Association for Computational Linguistics.

Zhiang Yue, Jingping Liu, Cong Zhang, Chao Wang, Haiyun Jiang, Yue Zhang, Xianyang Tian, Zhedong Cen, Yanghua Xiao, and Tong Ruan. 2023. Mamrc: A multi-answer machine reading comprehension dataset. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, page 2144–2148, New York, NY, USA. Association for Computing Machinery.

A Algorithm for Extracting Commonality Entities

The algorithm for extracting CE is shown in Algorithm 1. Specifically, for a given knowledge graph $G = \{V, E\}$, we first initialize its adjacency matrix M_G. Then, for each node $v \in V$, we count its 1-hop neighbor nodes and the types of edges connecting them. If node v is connected to a set of

	#train	#dev	average answer number	average context length	average question length
MultiSpanQA	5,230	658	2.89	279	10
MA-MRC (10k)	10,000	1000	2.31	77	10
QUOREF	1,963	215	2.45	431	19

Table 3: Dataset Statistic.

neighbor nodes $\bar{V}$ via edges of the same type, or if $\bar{V}$ point to v using edges of the same type, then $\bar{V}$ are considered as CE.

B Experimental Setup

B.1 MSQA Datasets

MultiSpanQA (Li et al., 2022) MultiSpanQA focuses on questions with more than one answer. The raw questions and contexts are extracted from the Natural Question dataset (Kwiatkowski et al., 2019).

MA-MRC (Yue et al., 2023) MA-MRC is a large-scale dataset containing over 100,000 questions, including both multi-span questions and single-span questions. In this work, we randomly sample 10,000 training data and 1,000 validation data and obtain **MA-MRC-10k** for our experiment.

QUOREF (Dasigi et al., 2019) The QUOREF dataset is sourced from Wikipedia and contains over 4,700 passages and more than 24,000 questions. The QUOREF dataset requires the model to possess certain co-reference resolution and reasoning abilities. In this work, we select questions with multiple answers for our experiment.

Since the official test sets of these datasets are not public, we report the performance on validation sets. Some statistics about the four datasets are shown in Table 3.

B.2 MSQA Models

Tagger (Li et al., 2022) Tagger utilizes BIO tags to label each token in context: the first token of the answer is labeled with "B", the other tokens of the answer are labeled with "I" and the tokens not in an answer are labeled with "O". In this work, we use *RoBERTa-base* [9] as the encoder.

SpanQualifier (Huang et al., 2023) : SpanQualifier enumerates all possible answer spans and obtains their corresponding confidence scores as correct predictions, then utilizes a learnable threshold to select the correct prediction spans. In this work, we also use RoBERTa-base as the encoder.

[9] https://huggingface.co/FacebookAI/roberta-base

Hyper-Parameter	Value (1st-tune)	Value (2nd-tune)
Learning Rate	3e-5	3e-5
Warmup Steps	100	100
Max Steps	15,000	8,000
Training Batch Size	32	8
Max Input Length	512	512
Max Output Length	64	64
Random Seed	1111	1111
Epochs	5	5
Optimizer	Adam	Adam

Table 4: Training Hyper-parameters. "1st-tune" and "2nd-tune" refer to the first step and the second step of the 2-step fine-tuning strategy, respectively.

BART (Lewis et al., 2020) and T5 (Raffel et al., 2020) : Both BART and T5 are pre-trained models with encoder-decoder architecture, which are commonly used in text generation tasks. In this work, we use the delimiter "#" to concatenate multiple answers.

B.3 QG Baselines

QAGen (Shakeri et al., 2020) QAGen uses a generative model to generate questions and answers. In this work, we fine-tune a *Llama-3.2-1B-Instruct* to generate questions and answers.

LIQUID (Lee et al., 2023) LIQUID first uses a summarization model and NER tools to extract named entities as candidate answers. Then, LIQUID employs a QG model to generate questions, and the questions and candidate answers are updated through multiple iterations. Lee et al. (2023) construct two synthetic datasets using Wikipedia and PubMed corpus. We refer to them as LIQUID-WIKI and LIQUID-MED, respectively [10].

[10] We download the LIQUID-WIKI and LIQUID-MED datasets from https://github.com/dmis-lab/LIQUID

B.4 Implementation Details

When converting passages to knowledge graphs, we utilize the LangChain *LLMGraphTransformer* [11] and invoke *GPT-4o-mini* [12]. When generating questions, we select *Llama-3.2-1B-Instruct* [13] and conduct Supervise Fine-Tuning (SFT) on MultiSpanQA and MA-MRC datasets. When selecting the optimal question, we select *UnifiedQA-T5-large*[14] and fine-tune it on MultiSpanQA and MA-MRC datasets. Training hyper-parameters are shown in Table 4.

C Additional Experiment Results

C.1 Main Results with REGULAR-MED

Table 5 shows the results of the 2-step fine-tuning experiment using the dataset constructed from PubMed corpus. We observe that REGULAR-MED achieves the best performance in most settings, which is consistent with the results in Table 1. Interestingly, despite the domain bias, the performance of the model trained with the REGULAR-MED dataset is quite close to the model trained with the REGULAR-WIKI dataset (for example, on the MultiSpanQA dataset, the Tagger model achieved EM F1 scores of 70.45 and 70.51, respectively). This suggests that during the pre-fine-tuning phase, the model primarily learns inductive reasoning abilities, and in the grained-fine-tuning phase, the model adapts to the domain of the downstream task.

C.2 Supervised Fine-Tuning Results

We utilize Llama(Grattafiori et al., 2024) and Qwen(Qwen et al., 2025) for our experiments. Specifically, we select Llama-3B[15], Llama-8B[16], Qwen2.5-3B[17], and Qwen2.5-7B[18], and conduct both In-Context Learning (ICL) (Brown et al., 2020) and LoRA (Hu et al., 2021) fine-tuning experiments. For the ICL experiments, we add 3 examples for each question; For the LoRA experiment, we set up two fine-tuning strategies: "Original" and "Merged." "Original" refers to fine-tuning with the original training data, while "Merged" refers to replacing 50% of the questions in the original training data with questions from the REGULAR-WIKI dataset.

The experimental results are shown in Table 6. It can be seen that replacing part of the training data with REGULAR-WIKI leads to some performance degradation. However, the results still outperform the ICL experiment, indicating that the REGULAR-WIKI dataset can be used to train LLMs and enhance their performance on QA tasks.

[11] https://python.langchain.com/
[12] https://openai.com/api/
[13] https://huggingface.co/meta-llama/Llama-3.2-1B-Instruct
[14] https://huggingface.co/allenai/unifiedqa-t5-large
[15] https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct
[16] https://huggingface.co/meta-llama/Llama-3.1-8B
[17] https://huggingface.co/Qwen/Qwen2.5-3B
[18] https://huggingface.co/Qwen/Qwen2.5-7B

D Definition of the Types of Question

Lee et al. (2023) proposes a category for question types based on the reasoning required to answer these questions, listed as follows:

- **Simple questions**: Questions simply derived from evidence texts with few lexical variations.
- **Lexical variation**: Questions created with lexical variations using synonyms and hypernyms.
- **Inter-sentence reasoning**: Questions that require high-level reasoning such as anaphora, or answers that are distributed across multiple sentences.
- **Number of answers**: Questions that specify the number of answers, which is a characteristic of a list of questions.
- **Entailment**: Questions that require textual entailment based on the evidence texts and commonsense.

E Analysis on REGULAR-MED Dataset

E.1 Number of Answers

We analyze the number of answers for each question in the REGULAR-MED datasets, as shown in Figure 7. The distribution of the number of answers in the REGULAR-MED dataset is quite similar to that of the REGULAR-WIKI dataset.

E.2 Types of Answers

We analyze the types of answers for each question in the REGULAR-MED datasets, as shown in Figure 7. The types of answers are different

	MultiSpanQA		MA-MRC		QUOREF	
	EM F1	PM F1	EM F1	PM F1	EM F1	PM F1
Tagger	68.58	83.62	79.69	87.50	66.77	81.78
+ QAGen-MED	67.04	82.89	79.48	87.19	68.86	82.20
+ LIQUID-MED	67.50	83.26	**80.31**	87.59	73.91	**85.67**
+ REGULAR-MED	**70.45**	**84.57**	80.27	**87.99**	**75.89**	84.77
SpanQualifier	71.58	83.50	79.85	86.83	62.21	75.36
+ QAGen-MED	69.81	83.26	57.68	73.74	58.29	73.02
+ LIQUID-MED	70.76	**84.52**	80.66	87.28	71.95	80.31
+ REGULAR-MED	**72.98**	83.19	**81.38**	**88.06**	**72.98**	**83.19**
BART-base	65.14	80.15	75.38	84.40	66.80	75.38
+ QAGen-MED	66.98	81.32	74.29	83.97	61.97	73.62
+ LIQUID-MED	66.67	81.14	75.84	84.60	67.24	77.17
+ REGULAR-MED	**68.98**	**82.59**	**75.88**	**85.28**	**67.44**	**78.51**
T5-base	69.24	82.93	78.39	86.21	65.47	76.40
+ QAGen-MED	69.81	83.26	78.89	86.32	61.41	73.25
+ LIQUID-MED	70.29	81.83	79.26	86.78	67.64	75.94
+ REGULAR-MED	**73.09**	**85.19**	**79.40**	**86.87**	**67.66**	**76.23**

Table 5: Additional Exact Match and Partial Match F1 scores of the MSQA models. The first line of each MSQA model refers to the original performance. "QAGen-MED" and "LIQUID-MED" refer to the 2-step fine-tuning baselines where models are first trained on the PubMed corpus's synthetic datasets. The best results are in **bold**.

	Llama3-3B		Llama3-8B		QWen2.5-3B		QWen2.5-7B	
	EM F1	PM F1	EM F1	PM F1	EM F1	PM F1	EM F1	PM F1
Zero-Shot	57.31	75.23	58.41	76.66	59.45	76.24	68.06	82.79
Few-Shot	64.98	80.06	68.73	84.13	65.48	79.90	70.64	84.56
SFT(Merged)	75.19	87.73	76.61	88.68	73.46	86.08	76.13	88.25
SFT(Original)	75.69	87.89	77.18	88.97	76.35	88.47	78.58	90.27

Table 6: Supervised Fine Tuning (SFT) on the MultiSpanQA dataset. We employ In-Context Learning (ICL) in the "Zero-Shot" and "Few-Shot" settings. "SFT(Merged)" refers to fine-tuning with LoRA using both the original training data and the synthetic data, while "SFT(Original)" refers to fine-tuning with LoRA using only the original training data.

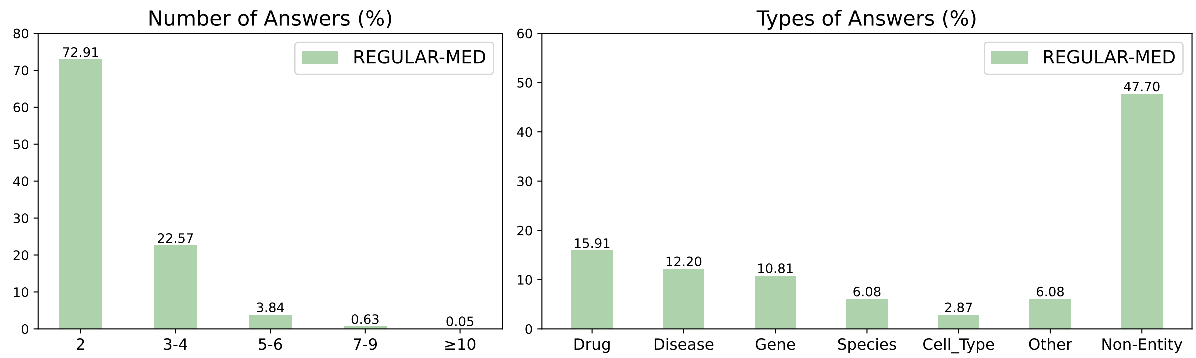

Figure 7: Left: Number of answers in the REGULAR-MED dataset; Right: Types of answers in the REGULAR-MED dataset. The numbers in the figures represent the percentage.

from the REGULAR-WIKI dataset. This is because the REGULAR-MED dataset is focused on the biomedical domain, so the extracted candidate answers are more likely to be specialized terms.

E.3 Types of Questions

We analyze the question type of each question in the REGULAR-MED datasets, as shown in Figure 8. The question type distribution in the REGULAR-MED dataset is also similar to that of REGULAR-WIKI, but the proportion of the "Number" type is higher. This may be because more numeric terms are included in the generated questions.

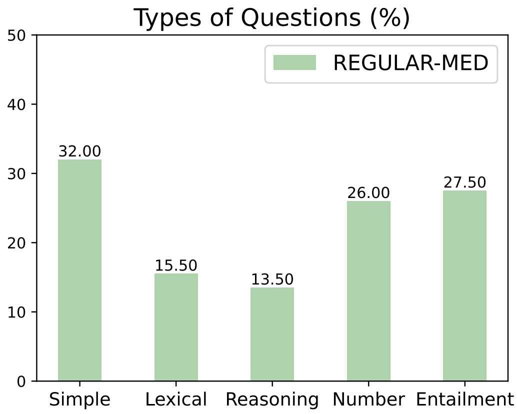

Figure 8: Types of questions in the REGULAR-MED dataset. The numbers in the figure represent the percentage.

Algorithm 1: Extracting Commonality Entities

Input: $G = \{V, E\}$: Knowledge Graph
Output: CE_list : Commonality Entities List
Function `ExtractCommonalityEntities`(G):
  $CE\_list \leftarrow \emptyset$;
  `/* Initialize adjacency matrix` M `of` G. `*/`
  $M \leftarrow$ adjacency_matrix(G);
  `/* Find commonality entites with the structure like` $B \leftarrow A \rightarrow C$ `or` $B \rightarrow A \leftarrow C$. `*/`
  foreach *entity* v *in* V **do**
    `/* Initialize` $Groups_1$ `and` $Groups_2$ `as a map. */`
    $Groups_1 \leftarrow map()$;
    $Groups_2 \leftarrow map()$;
    foreach *entity* u *in* V **do**
      `/* If there exists edge from` v `to` u`, then` $M[u][v] > 0$. `*/`
      $r_1 \leftarrow M[v][u]$;
      $r_2 \leftarrow M[u][v]$;
      if $r_1 > 0$ **then**
        $Groups_1[r_1] \leftarrow Groups_1[r_1] \cup m$;
      if $r_2 > 0$ **then**
        $Groups_2[r_1] \leftarrow Groups_2[r_1] \cup n$;
    foreach *group in* $Groups_1$ **do**
      `/* Add groups with more than 2 elements to` CE_list `*/`
      if $len(group) > 2$ **then**
        $CE\_list \leftarrow CE\_list \cup group$
    foreach *group in* $Groups_2$ **do**
      if $len(group) > 2$ **then**
        $CE\_list \leftarrow CE\_list \cup group$
  return CE_list;