RUBY: An Effective Framework for Multi-Constraint Multi-Hop Question Generation

Wenzhuo Zhao
School of Computer Science
South China Normal University
Guangzhou, China
wenzhuozhao@m.scnu.edu.cn

Shuangyin Li *
School of Computer Science
South China Normal University
Guangzhou, China
shuangyinli@scnu.edu.cn

Abstract

Inspired by theories in language psychology, it is natural to consider more constraints, such as intentions, logic, knowledge, etc., when a complex or multi-hop question is generated. As the subtask of Multi-Hop Question Generation (MHQG), the task of Multi-Constraint Multi-Hop Question Generation (MCHQG) is more aligned with human question theories. However, it is hard to determine how to bring various high-dimensional semantic constraints, and how to integrate each constraint across all hops when a multi-hop question is being generating. To address these challenges, we introduce an effective framework which includes constraint dimensionality reduction and divide-and-conquer-based dynamic projection; we call it RUBY. The proposed RUBY contains a module of high-dimensional semantic constraint dimension reduction and a module of sub-question answer pairs-based multi-hop question generation. Meanwhile, a Reasoning Dynamic Projection strategy is tailored to effectively incorporate the constraints into every hop of the multi-hop question. The experimental results demonstrate that RUBY consistently outperforms baseline models, which suggest that RUBY is able to effectively capture and integrate semantic constraints, leading to more accurate and human-like multi-hop question generation. We release the code and data to public[1].

1 Introduction

Question Generation (QG) embodies an aspect of Artificial General Intelligence (AGI). AGI requires that artificial intelligence not only answers questions, but also possesses the capability to ask questions. The ability to generate questions demonstrates AI's deep understanding of knowledge, proactive exploration, and innovative thinking, qualities at the core of AGI. As Einstein stated,

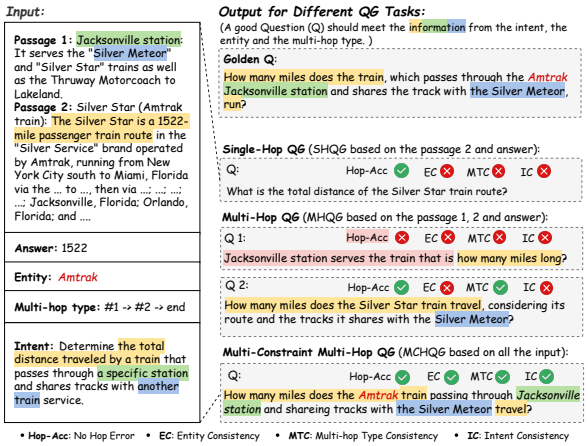


Figure 1: SHQG generates direct questions, while MHQG tends to produce more variable ones, increasing the chance of hop errors. MCHQG reduces randomness with added constraints, leading to more focused questions and fewer hop errors, as proven in the App. F.

"The formulation of a problem is often more essential than its solution." Thus, in an era of rapid AI advancement, teaching AI to ask questions will further propel the progress of AI science.

In the past, most of the research on Question Generation (QG) has focused on the generation of single-hop questions, which can be answered based on a single sentence or document (Sun et al., 2018; Kim et al., 2019; Wang et al., 2020; Fei et al., 2021). However, in recent years, as the demand for AI systems with more robust reasoning capabilities has increased, there has been growing interest in the task of multi-hop question generation (MHQG). MHQG requires aggregating the dispersed evidence from multiple paragraphs and generating answerable and factually coherent questions through reasoning (Su et al., 2020; Fei et al., 2022; Su et al., 2022; Xia et al., 2023; Hwang et al., 2024). These questions involve longer chains of reasoning compared to single-hop questions.

Theoretically, MHQG has primarily involved generating a question based on a single answer

*Corresponding author.
[1]https://github.com/zwz4949/MCHQG-RUBY

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 18164–18188
July 27 - August 1, 2025

across multiple passages. However, this way typically resulted in questions being generated randomly, lacking directionality. Drawing on theories from psycholinguistics, Kormos (2006) noted that humans start with a conceptualization stage to obtain a preverbal plan **BEFORE** language production. Take human questioning as an example. In the conceptualization stage, the entities, intentions, logic, concepts, etc., can be treated as the multiple constraints to guide the question generation process. This suggests that humans plan the messages of questions rather than posing them randomly. Therefore, starting with multiple constraints can make multi-hop question generation more aligned with human questioning theory by providing directionality. It also sets a preliminary logical framework for the multi-hop logic chain of the questions to be generated, potentially reducing the likelihood of errors in multi-hop reasoning.

The task of multi-hop question generation that begins with the process of conceptualization is called Multi-Constraint Multi-Hop Question Generation (MCHQG), which requires not only the answer and the passages, but also high-dimensional and low-dimensional semantic constraints to guide the planning of messages when generating a multi-hop question.

However, although MCHQG enables directional questioning, it still faces two major challenges. Firstly, **how can high-dimensional semantic constraints be effectively brought in?** For example, as shown in Figure 1, the given intent serves as a high-dimensional semantic constraint. In order to meet this intent, the generated question must inquire about the "distance traveled by a train", incorporating specific information from the passages about "a specific station" and "another train", with a logical connection between these elements. Secondly, **how can multiple constraints be effectively integrated into each hop of a multi-hop question?** It is natural that not every hop necessarily needs to incorporate all the constraints, which means that it is adaptive to integrate the constraints into each hop. Although there are some existing studies on constrained question generation (Cheng et al., 2021; Uto et al., 2023; Liu et al., 2023; Li and Zhang, 2024), these works either focus on generating single-hop questions or apply only a single constraint. Thus, they are difficult to apply effectively to scenarios that require both multi-constraint and multi-hop question generation.

Therefore, how to adaptively incorporate multiple types of high-dimensional semantic constraints into hops is essential and challenging when generating questions with complex logical chains.

To address these challenges, we propose an effective framework named **RUBY**, which incorporates constraint **R**eduction and divide-and-conq**U**er-**B**ased d**Y**namic Projection. Specifically, we introduce a high-dimensional constraint reduction module, which includes passage sorting and multi-hop skeleton construction, simplifying high-dimensional constraints into low-dimensional ones and streamlining the reasoning process. To leverage these constraints more effectively, we adopt a divide-and-conquer approach, breaking down multi-hop question generation into sub-QA pairs (SQAG), which are then integrated into a coherent multi-hop question (MHQG). Additionally, we propose Reasoning Dynamic Projection, a strategy that dynamically incorporates various constraints into reasoning and question generation. This is achieved by calculating weights for all constraints at each decoding step in both SQAG and MHQG.

Our main contributions are as follows:

- We propose RUBY, a framework for the MCHQG task, which utilizes LLMs and a divide-and-conquer strategy to generate high-quality multi-hop questions.
- We propose a high-dimensional semantic constraint dimension reduction module to effectively utilize high-dimensional semantic constraints.
- We introduce a strategy, Reasoning Dynamic Projection, to reasonably incorporate the constraints into the multi-hop reasoning and question generation process.

2 Related Work

Question Generation (QG) is vital for educational systems (Jia et al., 2021; Zhao et al., 2022) and enhances QA tasks by improving reasoning capabilities (Duan et al., 2017; Tang et al., 2017). Early QG models were rule-based (Mostow and Chen, 2009; Heilman, 2011; Chali and Hasan, 2012), limiting their scalability and diversity. The seq2seq QG models (Zhao et al., 2018) overcame these limitations, enabling more diverse question generation. However, these models, trained on single-hop datasets like SQuAD (Rajpurkar et al., 2016) and NewsQA (Trischler et al., 2017), generate simple, single-hop questions. Recently, there has been

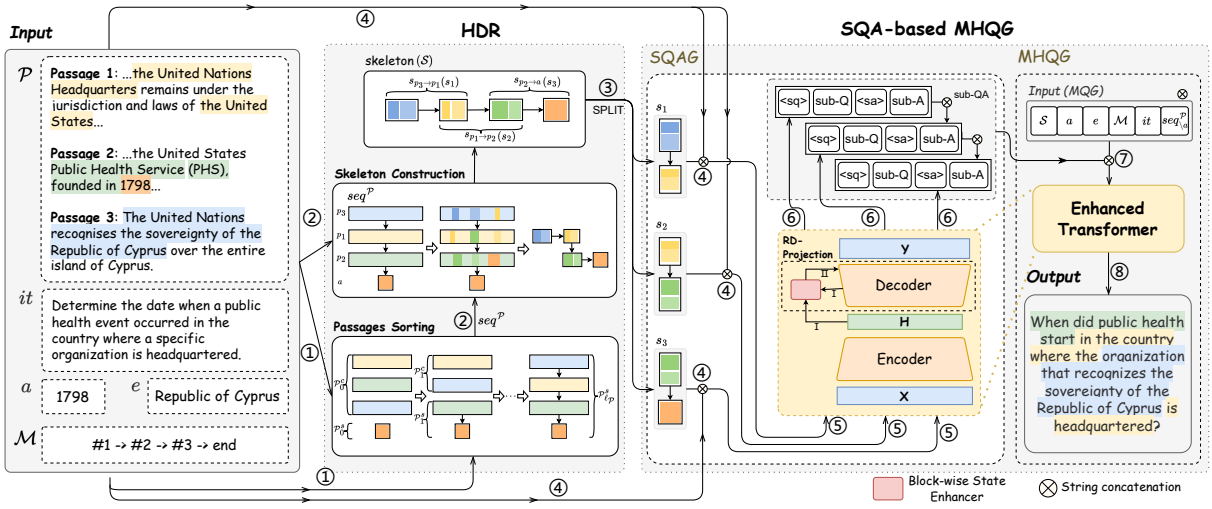


Figure 2: The Overview of the proposed RUBY. ① to ⑧ shows the steps of the RUBY pipeline. HDR including the process of passage sorting and the construction of multi-hop skeleton conducts a dimension reduction for high-dimensional semantic constraints. SQA-based MHQG generates and fuses the sub-QA pairs into a multi-hop question using an enhanced transformer with RD-Projection.

increasing interest in tackling multi-hop question generation tasks.

Multi-Hop Question Generation (MHQG). Previous work on MHQG has focused on generating questions from multiple paragraphs with a single answer. Su et al. (2020), Pan et al. (2020), and Fei et al. (2022) employed GNN-based (Li et al., 2025) methods over semantic, entity, and attention graphs, respectively, while Su et al. (2022) incorporated question-answering techniques. Xia et al. (2023) used multi-level content planning to enhance decoding; Hwang et al. (2024) increased complexity via sequential rewriting; and Lin et al. (2024) introduced chain-of-thought reasoning. However, these methods are tailored to MHQG, and their effectiveness in MCHQG remains unexplored. In this work, we assess their applicability in the MCHQG setting.

Constrained Question Generation. Constraining specific aspects of question generation has important applications, such as in intelligent tutoring systems (Guo et al., 2024). Cao and Wang (2021) defined question type ontologies for type-aware generation, while Cheng et al. (2021) proposed a graph-based framework to control question difficulty by reasoning steps. Bi et al. (2021); Srivastava and Goodman (2021) generate questions based on various features, and Liu et al. (2023) focus on entity-centered generation. Li and Zhang (2024) use LLMs for planning questions, enhancing control over content and difficulty. However, these studies mainly focus on single-hop questions or a single constraint, limiting their applicability to MCHQG.

3 Problem Definition

The goal of multi-constraint multi-hop question generation (MCHQG) is to generate multi-hop questions based on several constraints[2]. Given constraints such as an answer a, an entity e, a multi-hop type $\mathcal{M}$, an intent it, and a set of passages $\mathcal{P} = \{p_i\}_{i=1}^{\ell}$, the task is to generate a multi-hop question $\hat{y}$. Here, ℓ denotes the number of passages. The question is generated by the model f_{G}, which takes the constraints as input:

$$\hat{y} = f_{\text{G}}(a, e, \mathcal{M}, it, \mathcal{P}). \quad (1)$$

The multi-hop type $\mathcal{M} = (\mathcal{V}, \mathcal{E})$ represents the logical structure of a question. It consists of nodes $\mathcal{V} = \mathcal{V}_\ell \cup \{v_{\text{end}}\}$ and edges $\mathcal{E} = \{(v_c, v_d) \mid v_c, v_d \in \mathcal{V}_\ell \text{ and } v_c \neq v_d\} \cup \{(v_\ell, v_{\text{end}})\}$. Each node $v \in \mathcal{V}_\ell$ corresponds to a passage $p \in \mathcal{P}$, and the final node v_{end} corresponds to the answer a. For example, in Figure 3, nodes $v_1, v_2, v_3, v_{\text{end}}$ correspond to passages p_3, p_1, p_2, and the answer a. The relationships are represented as $(p_3, p_1), (p_1, p_2), (p_2, a)$.

4 Methodology

4.1 Overview

The proposed framework RUBY consists of two primary modules: High-dimensional semantic con-

[2]The detailed explanations of the terminology are presented in Table 16.

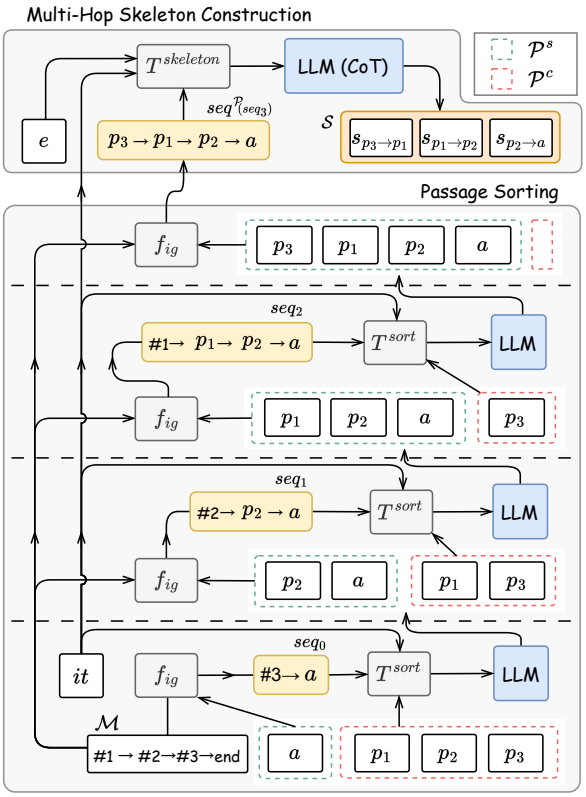


Figure 3: The module of High-Dimensional Constraint Dimension Reduction, with an example illustrating the iterative passage sorting process and the construction of the multi-hop skeleton.

straint Dimension Reduction (HDR) and Sub-Question Answer pair-based Multi-Hop Question Generation (SQA-based MHQG), as illustrated in Figure 2.

The HDR module has two stages: Passage Sorting and Multi-hop Skeleton Construction. In passage sorting, a sorted sequence of passages, $seq^{\mathcal{P}} = [p'_1, p'_2, \ldots, p'_\ell, a]$, is generated through an iterative sorting process, considering the multi-hop type $\mathcal{M}$, the intent it, and the passages $\mathcal{P}$. Here, p'_i denotes the passage corresponding to the node #i in $\mathcal{M}$, and a is treated as a passage, and the sorted sequence of passages excluding a can be represented as $seq^{\mathcal{P}}_{\setminus a} = seq^{\mathcal{P}} \setminus [a] = [p'_1, p'_2, \ldots, p'_\ell]$. In the multi-hop skeleton construction module, the sequence $seq^{\mathcal{P}}$ generated from passage sorting is further refined. Using the intent it, the entity e, the answer a, and a Chain-of-Thought approach (CoT), essential information is captured from the passages to create a multi-hop skeleton $\mathcal{S}$. This skeleton can be split into ℓ parts based on $\mathcal{M}$ and $seq^{\mathcal{P}}$, and expressed as:

$$\mathcal{S} = [s_{p'_1 \rightarrow p'_2}, s_{p'_2 \rightarrow p'_3}, \ldots, s_{p'_\ell \rightarrow a}] = [s_1, s_2, \ldots, s_\ell]. \quad (2)$$

The SQA-based MHQG module consists of two stages: SQAG for sub-question-answer pair generation, followed by MHQG for multi-hop question generation. Both stages utilize Reasoning Dynamic Projection (RD-Projection). In SQAG, sub-question-answer pairs $\mathcal{Z} = [z_1, z_2, \ldots, z_\ell]$ are generated from $\mathcal{S}$ and $seq^{\mathcal{P}}_{\setminus a}$. These pairs are then combined with inputs including $\mathcal{S}$, it, e, a, $\mathcal{M}$, and $\mathcal{P}$ to form the input for MHQG, where a multi-hop question y is generated.

4.2 High-Dimensional Semantic Constraint Dimension Reduction

High-dimensional semantic constraint dimension reduction (HDR) takes $\mathcal{P}$, it, $\mathcal{M}$, a, and e as input and outputs the multi-hop skeletons $\mathcal{S}$. Given the powerful reasoning ability of LLMs, we first use LLM to sort the passages and then construct the multi-hop skeletons based on the designed prompts.

Passage Sorting. In this stage, we mainly perform dimension reduction for the multi-hop type. Since multi-hop relationships are typically sequential, we first sort the passages $\mathcal{P}$. To guide the LLM in passage sorting, we use a specifically designed prompt, as detailed in Table 18. The algorithm of passage sorting is shown in App. I.

Due to the intricate interactions between $\mathcal{P}$ and other available information, we perform passage sorting iteratively rather than in a single round. Initially, a is set as the first passage in the sorted sequence of passages seq_0, and then we iterate over the remaining passages for ℓ rounds.

At the i-th iteration ($1 \leq i \leq \ell$), we define $\mathcal{P}' = \mathcal{P} \cup \{a\}$ and divide it into two subsets: candidate passages $\mathcal{P}^c_i$ and selected passages $\mathcal{P}^s_i$. To generate the sorted sequence at iteration i, we first generate the sequence seq_{i-1} using the function f_{ig}, which replaces the corresponding nodes in $\mathcal{M}$ with the previously selected passages $\mathcal{P}^s_{i-1}$. Then, based on it and seq_{i-1}, we select one passage p^*_i from $\mathcal{P}^c_{i-1}$ using the LLM with the prompt template T^{sort}. Afterward, we update $\mathcal{P}^s_i = \mathcal{P}^s_{i-1} \cup \{p^*_i\}$ and $\mathcal{P}^c_i = \mathcal{P}' \setminus \mathcal{P}^s_i$, ensuring that $\mathcal{P}^c_i \cap \mathcal{P}^s_i = \emptyset$. Formally, the selected passage p^*_i at the i-th iteration can be generated as:

$$p^*_i = \text{LLM}(T^{\text{sort}}(it, f_{ig}(\mathcal{M}, \mathcal{P}^s_{i-1}), \mathcal{P}^c_{i-1})). \quad (3)$$

The passage corresponding to #i in $\mathcal{M}$ can be expressed as $p'_i = p^*_{\ell+1-i}$. Finally, the sorted sequence of passages and the sequence excluding the answer are obtained respectively as:

$$seq^{\mathcal{P}} = seq_\ell = f_{ig}(\mathcal{M}, \mathcal{P}^s_\ell), seq^{\mathcal{P}}_{\setminus a} = seq^{\mathcal{P}} \setminus [a]. \quad (4)$$

Multi-Hop Skeleton Construction. After ob-

taining the sorted sequence of passages from passage sorting, we perform dimension reduction on the intent, while further reducing the dimensionality of the multi-hop type. Using it, a, $seq^{\mathcal{P}}$, e, and the prompt template $T^{skeleton}$, we construct the LLM input as $T^{skeleton}(it, a, seq^{\mathcal{P}}, e)$, where $seq^{\mathcal{P}}$ is the result of the initial dimension reduction for the multi-hop type. A CoT approach is applied to analyze the inputs and extract key information for constructing the multi-hop skeleton $\mathcal{S}$. This process refines $\mathcal{M}$ and the abstract intent it into precise, passage-derived information, forming a multi-hop skeleton aligned with the intent and multi-hop type. The construction of $\mathcal{S}$ can be expressed as:

$$\mathcal{S} = \text{LLM}(T^{skeleton}(it, a, seq^{\mathcal{P}}, e)). \tag{5}$$

4.3 SQA-based MHQG

Given e, a, it, $seq^{\mathcal{P}}_{\backslash a}$, and $\mathcal{S}$, the goal of SQA-based MHQG is to generate a multi-hop question y. As shown in Figure 2, we decompose the multi-hop question generation process into sub-processes for generating sub-QA pairs $\mathcal{Z}$. Each sub-QA pair z_i is generated from an individual sub-process. Then, $\mathcal{Z}$ is used to generate the multi-hop question y, as expressed below:

$$p(y|e, a, it, seq^{\mathcal{P}}_{\backslash a}, \mathcal{S}) =$$
$$\mathbb{E}_{\mathcal{Z}}\Big[\underbrace{p(y|\mathcal{Z}, \mathcal{M}, e, a, it, \mathcal{S}, seq^{\mathcal{P}}_{\backslash a})}_{\text{MHQG}} \cdot \underbrace{\prod_{i=1}^{\ell} p(z_i|e, it, s_i, p'_i)}_{\text{SQAG}}\Big], \tag{6}$$

where each sub-QA pair z_i is generated based on p'_i from $seq^{\mathcal{P}}_{\backslash a}$, s_i from $\mathcal{S}$, and e and it.

The primary model for both SQAG and MHQG is a Transformer enhanced with the Reasoning Dynamic Projection (RD-Projection) strategy. For SQAG, the input is $f_c(e, it, s_i, p'_i)$, where f_c denotes the string concatenation function, and the output is z_i. For MHQG, the input is $f_c(\mathcal{Z}, e, a, it, \mathcal{S}, seq^{\mathcal{P}}_{\backslash a})$, where $\mathcal{Z} = [z_1, \ldots, z_\ell]$ represents the sub-QA pairs obtained from SQAG, and the output is the multi-hop question y. Notably, the Transformer used for SQAG does not share weights with the one used for MHQG.

Formally, given the input sequence x, we divide it into several blocks $[b_1, \ldots, b_j, \ldots, b_L]$, where L denotes the number of blocks. For SQAG, b_j in x is equal to one of e, it, s_i, or p'_i. For MHQG, b_j is equal to one of z_i, e, it, s_i, p'_i, or a. Here, j represents the index of a block. A special token st_j is placed before each block b_j to separate the blocks and integrate their semantic information. Consequently, the input sequence x is reconstituted as $[st_1, b_1, \ldots, st_j, b_j, \ldots, st_L, b_L]$.

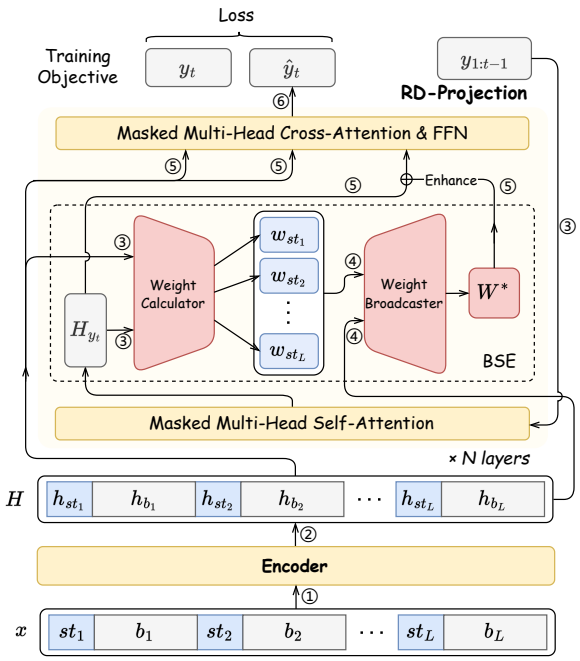


Figure 4: The enhanced Transformer with Reasoning Dynamic Projection (RD-Projection), which conducts block-wise projection for the constraints. ① to ⑥ shows the steps. BSE indicates the block-wise state enhancer.

Encoder. We first use a pretrained Transformer encoder to encode the input sequence x, generating the representations:

$$\mathbf{H} = \text{Encoder}(x) = [\mathbf{h}_{st_1}, \mathbf{h}_{b_1}, \mathbf{h}_{st_2}, \mathbf{h}_{b_2}, \ldots, \mathbf{h}_{st_L}, \mathbf{h}_{b_L}], \tag{7}$$

where $\mathbf{H} \in \mathbb{R}^{|x| \times d}$, $|x|$ is the length of x, and d is the dimension of the representations.

Reasoning Dynamic Projection. Reasoning Dynamic Projection (RD-Projection) is an enhanced Transformer decoder that integrates a Block-wise State Enhancer. As illustrated in Figure 4, RD-Projection generates the next token y_t by utilizing the previously generated tokens $y_{1\ldots t-1}$. The self-attention mechanism is applied to integrate the features of $y_{1\ldots t-1}$:

$$\mathbf{H}_{y_t} = g_s(y_{1\ldots t-1}) \in \mathbb{R}^{1 \times d}, \tag{8}$$

where g_s denotes the self-attention layer. To incorporate the block representations from the encoder, we introduce the Block-wise State Enhancer module, which precedes the cross-attention layer. This module enhances the representations $\mathbf{H}^*_{y_t}$, allowing the cross-attention mechanism to more effectively interact with the block features.

Block-wise State Enhancer. The block-wise

state enhancer consists of two components: Weight Calculator and Weight Broadcaster. The weight calculator computes the block-wise weights based on $\mathbf{H}$ and $\mathbf{H}_{y_t}$. For each block b_j in x, the block-wise weight w_{st_j} is calculated by:

$$w_{st_j} = f_w([\mathbf{H}_{y_t}; \mathbf{h}_{st_j}]), \tag{9}$$

where $[;]$ denotes concatenation along the representation dimension, and f_w is a two-layer head for weight computation. The block-wise weights for all blocks are denoted as $\mathbf{W}_{st} = \{w_{st_j}\}_{j=1}^{L}$.

To project the block-specific weight onto each token within its corresponding block b_j, we assign w_{st_j} to all tokens in the block and set weights for tokens outside these blocks to $-\infty$. The initial weight projection sequence is described as follows:

$$\begin{aligned}\mathbf{W}_{init,p} = [&-\infty, w_{st_1}, w_{st_1}, \ldots, w_{st_1}, \ldots, \\ &-\infty, w_{st_L}, w_{st_L}, \ldots, w_{st_L}, \ldots, -\infty].\end{aligned} \tag{10}$$

Next, the weight projection sequence is refined as follows:

$$\begin{aligned}\mathbf{W}_p = f_s(\mathbf{W}_{init,p}) = [&0, w'_{st_1}, w'_{st_1}, \ldots, w'_{st_1}, \ldots, \\ &0, w'_{st_L}, w'_{st_L}, \ldots, w'_{st_L}, \ldots, 0],\end{aligned} \tag{11}$$

where f_s denotes the softmax function. The enhanced block-wise representations W^* are then computed as:

$$\mathbf{W}^* = \mathbf{W}_p\mathbf{H} \in \mathbb{R}^{1\times d}. \tag{12}$$

Finally, the output of RD-Projection is obtained by:

$$\hat{y}_t = g_{c,f}(\tilde{\mathbf{Q}}, \mathbf{K}, \mathbf{V}), \quad \tilde{\mathbf{Q}} = (\mathbf{Q} + \mathbf{W}^*)\mathbf{\Theta}, \tag{13}$$

where $\mathbf{\Theta} \in \mathbb{R}^{d\times d}$ is an MLP layer, $g_{c,f}$ represents the cross-attention layer and feed-forward network, and $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ correspond to $\mathbf{H}_{y_t}$, $\mathbf{H}$, $\mathbf{H}$.

4.4 Training

The training process consists of two stages: first, training SQAG, and second, training MHQG. Each stage has distinct input-output pairs and model parameters. The algorithm for the training process is shown in App. J.

For SQAG, let $\mathcal{D}_s = \left\{(x^{(k_s)}, y^{(k_s)})\right\}_{k_s=1}^{|\mathcal{D}_s|}$, where the input $x^{(k_s)} = (e, it, s_{k_s}, p_{k_s})$ and the output $y^{(k_s)}$ is a sub-QA pair. The model parameters are ξ_s, and the loss function is:

$$\mathcal{L}_s = -\sum_{k_s=1}^{|\mathcal{D}_s|}\sum_{l_s=1}^{|y^{(k_s)}|} \log P_s\left(y_{l_s}^{(k_s)} \middle| x^{(k_s)}, y_{<l_s}^{(k_s)}; \xi_s\right), \tag{14}$$

where P_s is the conditional log-likelihood for generating $y_{l_s}^{(k_s)}$ given $x^{(k_s)}$ and preceding tokens.

For MHQG, let $\mathcal{D}_m = \left\{(x^{(k_m)}, y^{(k_m)})\right\}_{k_m=1}^{|\mathcal{D}_m|}$, where $x^{(k_m)} = (\mathcal{Z}, \mathcal{M}, e, a, it, \mathcal{S}, seq_{\setminus a}^{\mathcal{P}})$, and the output $y^{(k_m)}$ is a multi-hop question. The model parameters are ξ_m, and the loss function is:

$$\mathcal{L}_m = -\sum_{k_m=1}^{|\mathcal{D}_m|}\sum_{l_m=1}^{|y^{(k_m)}|} \log P_m\left(y_{l_m}^{(k_m)} \middle| x^{(k_m)}, y_{<l_m}^{(k_m)}; \xi_m\right), \tag{15}$$

where P_m is the conditional log-likelihood for generating $y_{l_m}^{(k_m)}$ given $x^{(k_m)}$ and preceding tokens.

5 Experiments

5.1 Experimental Setup

Datasets. We conduct experiments on two challenging multi-hop QA datasets: MuSiQue (Trivedi et al., 2022) and HotpotQA[3] (Yang et al., 2018), which contain 2- to 4-hop questions and 2-hop questions, respectively. Since the test sets for the datasets are not publicly available, we use part of the original training sets as the validation sets and the original validation sets as the test sets. Due to the multi-hop nature and the introduction of additional constraints in the MCHQG task, we conduct a data-processing workflow. The detailed process is provided in App. K.

Metrics. We mainly measured BLEU (B) (Papineni et al., 2002), METEOR (MTR) (Banerjee and Lavie, 2005), ROUGE-L (R-L) (Lin, 2004), and BERTScore (BSc) (Zhang* et al., 2020) scores following previous studies.

Baselines. For general pretrained-language models based methods, we evaluate our proposed framework RUBY against several strong baselines designed for MHQG, including MulQG (Su et al., 2020), CQG (Fei et al., 2022), E2EQR (Hwang et al., 2024), and MultiFactor (Xia et al., 2023). Additionally, we compare RUBY with several general pretrained language models, such as T5-base (Raffel et al., 2020), FLAN-T5-base (Longpre et al., 2023), BART-base (Lewis et al., 2019), and MixQG-base (Murakhovs'ka et al., 2022). To explore the capabilities of large language models (LLMs) in MCHQG, we finetune two LLMs as baselines: Llama-3.1-8B-Instruct (Dubey et al., 2024) and Mistral-7B-Instruct-v0.3 (Jiang et al., 2023), using QLora (Dettmers et al., 2024) with 4-bit quantization.

5.2 Main Results

The performance of the baselines and RUBY on MuSiQue is shown in Table 1 with the following main insights. RUBY significantly outperforms other pre-trained seq2seq baselines and LLMs. No-

[3]All results on HotpotQA are in App. B.

Model	B-1	B-2	B-3	B-4	R-L	MTR	BSc
Llama-3.1-8B†	55.31	41.41	32.45	26.05	48.75	29.27	56.11
Mistral-7B-v0.3†	54.77	41.20	32.51	26.26	48.18	28.92	55.84
BART-base†	54.71	41.93	33.43	27.14	50.90	<u>31.39</u>	57.93
T5-base†	55.75	42.73	34.00	27.59	50.86	30.87	57.50
FLAN-T5-base†	<u>56.98</u>	<u>43.60</u>	<u>34.75</u>	<u>28.32</u>	50.81	29.75	57.17
MixQG-base†	55.05	42.26	33.77	27.50	<u>51.57</u>	31.07	<u>58.03</u>
MulQG	35.46	21.52	14.37	10.04	31.32	17.65	24.97
CQG	35.16	23.06	16.20	11.62	38.00	23.21	34.44
E2EQR	50.84	37.96	30.00	24.40	44.49	29.20	55.33
MultiFactor	54.89	41.82	33.24	27.02	49.97	30.46	56.65
RUBY	**58.64**	**45.45**	**36.59**	**30.04**	**52.89**	**31.50**	**59.40**

Table 1: Automatic evaluation on MuSiQue. The **Bold** and <u>underline</u> mark the best and second-best results. The models with † are fine-tuned foundational models.

tably, RUBY's use of the divide-and-conquer approach can also be considered a multi-level content planning-enhanced process, further strengthened by high-dimensional semantic constraint dimension reduction. This enables RUBY to surpass the performance of another multi-level content planning-enhanced baseline, MultiFactor, demonstrating that HDR is an effective module. Additionally, although common pre-trained models have fewer parameters, they can outperform LLMs in terms of performance. Furthermore, the performance of E2EQR and MultiFactor is inferior to that of their backbone models, BART-base and MixQG-base, respectively. This suggests that the enhancement modules used in MHQG do not generalize well to MCHQG. Overall, these findings validate that RUBY is more effective in MCHQG than other baselines.

5.3 Constraints Consistency Evaluation

We evaluate how effectively the model-generated questions satisfy the additional constraints introduced by MCHQG, focusing on entity consistency (EC), intent consistency (IC), and multi-hop type consistency (MTC) on MuSiQue. The results are shown in Table 2. For detailed evaluation metrics, refer to App. E.

The experimental results show that RUBY achieves leading performance in intent consistency and multi-hop type consistency, along with strong results in entity consistency. Furthermore, while LLMs perform worse than general pre-trained baselines on conventional automatic metrics, they surpass these baselines in IC and MTC. This highlights the ability of LLMs to process high-dimensional semantic constraints and construct a logical multi-hop skeleton for reasoning.

Model	EC	IC	MTC
Llama-3.1-8B	0.8413	<u>4.1032</u>	<u>3.8903</u>
Mistral-7B-v0.3	0.8654	3.9889	3.7982
BART-base	**0.9373**	3.5931	3.4422
T5-base	0.8948	3.6584	3.4082
FLAN-T5-base	0.8824	3.7080	3.5408
MixQG-base	0.8759	3.7897	3.5996
MulQG	0.3429	1.6241	1.6432
CQG	0.3011	1.9306	1.8986
E2EQR	0.8700	3.5963	3.3867
MultiFactor	0.9019	3.7289	3.5572
RUBY	<u>0.9033</u>	**4.1130**	**3.8955**

Table 2: Comparison of models based on the constraint consistency.

Overall, RUBY excels in maintaining constraint consistency by effectively bringing in high-dimensional semantics and integrating multiple constraints into the question generation process.

5.4 Ablation Studies

To evaluate the effectiveness of RUBY, we create three variants by removing key components: HDR, RD-Projection, and SQAG on MuSiQue, as shown in Table 3 on the MuSiQue dataset.

RUBY w/o HDR. This variant removes the high-dimensional semantic constraint reduction module and does not utilize multi-hop skeletons or sorted sequences of passages for SQAG and MHQG.

RUBY w/o SQAG. Eliminates the divide-and-conquer approach by removing the SQAG stage. In this variant, MHQG directly uses the intent, entity, answer, multi-hop type, and passages.

RUBY w/o RD-Projection (RD-P). Replaces

Model	B-4	R-L	BSc	EC	IC	MTC
RUBY	**30.04**	**52.89**	**59.40**	**0.9033**	**4.1130**	**3.8955**
w/o RD-P	29.14 (↓ 3.00%)	52.45 (↓ 0.83%)	59.06 (↓ 0.57%)	0.8890 (↓ 1.58%)	3.9765 (↓ 3.32%)	3.8478 (↓ 1.23%)
w/o SQA	27.79 (↓ 7.49%)	51.25 (↓ 3.10%)	58.18 (↓ 2.06%)	0.8792 (↓ 2.67%)	3.9504 (↓ 3.95%)	3.7577 (↓ 3.54%)
w/o HDR	28.22 (↓ 6.06%)	51.67 (↓ 2.31%)	58.15 (↓ 2.13%)	0.8916 (↓ 1.31%)	3.8909 (↓ 5.41%)	3.6741 (↓ 5.69%)

Table 3: Ablation studies on MuSiQue dataset, including conventional automatic metrics (B-4, R-L, BSc) and constraint consistency metrics (EC, IC, MTC).

Model	B-4	R-L	BSc
		2-hop	
MixQG	36.84	58.21	66.38
RUBY	**38.61** (↑ 4.81%)	**59.10** (↑ 1.54%)	**67.52** (↑ 1.73%)
		3-hop	
MixQG	23.77	47.27	52.52
RUBY	**27.03** (↑ 13.72%)	**49.17** (↑ 4.02%)	**54.42** (↑ 3.61%)
		4-hop	
MixQG	18.65	40.68	44.46
RUBY	**21.20** (↑ 13.67%)	**42.09** (↑ 3.47%)	**45.52** (↑ 2.38%)

Table 4: Performance comparison of RUBY and MixQG-base across different hops.

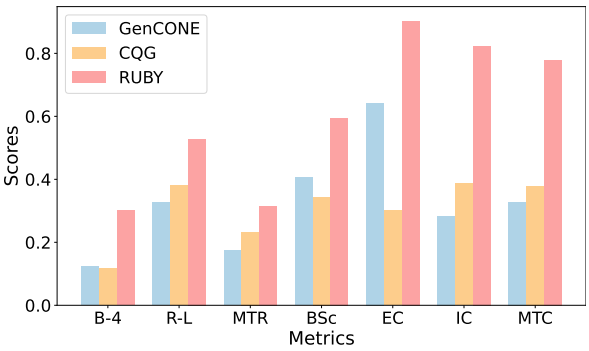


Figure 5: The performance comparison of single-hop single-constraint QG models, multi-hop QG models, and multi-constraint multi-hop QG models. For clarity, IC and MTC scores are scaled down by dividing by 5, and B-4, R-L, MTR, and BSc scores are divided by 100.

the reasoning dynamic projection with a standard transformer decoder. The cross-attention layer's query (Q) is no longer enhanced during SQAG and MHQG.

Removing HDR results in a significant drop in IC and MTC, underscoring the importance of dimensionality reduction for high-dimensional semantic constraints.

Removing SQAG reduces performance across all metrics, demonstrating the value of the divide-and-conquer approach. The SQAG stage generates sub-QA pairs that enhance control and precision in question generation.

Finally, removing RD-Projection leads to consistent performance drops, underscoring the importance of dynamically projecting weights to constraints during decoding. This strategy is essential for precise token selection and seamless hop connections within a question.

5.5 Performance of Different-Hop QG

We evaluated the performance of models on 2-hop, 3-hop, and 4-hop question generation using MuSiQue. As shown in Table 4, RUBY consistently outperforms its backbone model, MixQG-base, across all hop levels.

For 2-hop questions, RUBY achieves improvements of 4.81%, 1.54%, and 1.73% in BLEU-4, ROUGE-L, and BERTScore, respectively. The performance gap widens for 3-hop questions, with RUBY demonstrating increases of 13.72% in BLEU-4, 4.02% in ROUGE-L, and 3.61% in BERTScore. This trend continues for 4-hop questions, where RUBY delivers substantial gains of 13.67% in BLEU-4, 3.47% in ROUGE-L, and 2.38% in BERTScore. These results highlight RUBY's ability to generate high-quality questions, particularly for more complex multi-hop questions, with its performance advantage growing as question complexity increases. The performance of other baselines on different-hop question generation is further detailed in App. C.

5.6 Performance of the Single-Constraint Single-Hop QG Model on MCHQG

We select the open-source model GenCONE (Liu et al., 2023), which is capable of generalizing to MCHQG and is designed for single-constraint (entity-centric), single-hop question generation, to conduct experiments using MuSiQue. As shown in Figure 5, while GenCONE performs comparably to CQG on conventional automatic evaluation metrics, it surpasses CQG by up to 113.5% in entity consistency (EC). This result underscores the effectiveness of GenCONE's design for generating entity-constrained questions.

However, GenCONE falls significantly behind CQG in intent consistency (IC) and multi-hop type consistency (MTC). This is largely because GenCONE is tailored for single-hop question generation based on only one passage, limiting its ability to process and integrate broader contextual informa-

tion. Furthermore, while GenCONE is optimized for entity-centric tasks, it underperforms compared to RUBY across all metrics, including EC. This shortfall arises from GenCONE's inability to incorporate and reconcile multiple constraints effectively in multi-hop question generation tasks. This highlights the inherent challenges of adapting traditional single-constraint single-hop QG models to scenarios that demand both multi-constraint and multi-hop question generation.

6 Conclusion

We present the task of Multi-Constraint Multi-Hop Question Generation (MCHQG), drawing inspiration from theories in language psychology. To tackle the challenges of bringing in high-dimensional semantic constraints and effectively integrating multiple constraints into multiple hops, we propose RUBY, a robust and effective framework. Experimental results, evaluated through both conventional automatic metrics and constraint consistency metrics, demonstrate that RUBY surpasses baseline models, delivering superior performance in both metrics.

Limitations

Our work has some limitations. Firstly, RUBY does not specifically enhance entity consistency. We believe that future work on MCHQG should focus on ensuring consistency across all constraints, which could further enhance performance. Secondly, RUBY has been tested on English multi-hop question-answer datasets but has not been evaluated on datasets in other languages.

Ethics Statement

RUBY aims to improve the performance in the MCHQG task. Our question generation is within the scope of the dataset. In our work, we strictly adhered to the ethical guidelines established by the academic and open-source communities. All datasets and models used in this study are publicly available. We did not collect new datasets but instead conducted experiments and constructed additional constraints based on these existing works. There is minimal bias introduced in the experiments because the seed is fixed throughout the entire experiment. There are no conflicts of interest among the authors. Thus, there is no ethical issue in our work.

Acknowledgments

This work was supported by National Key Research and Development Program of China (2020YFA0712500), Major Program of National Language Committee (WT145-39) and National Natural Science Foundation of China (No. 62006083).

References

Gaurav Arora, Shreya Jain, and Srujana Merugu. 2024. Intent detection in the age of LLMs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1559–1570, Miami, Florida, US. Association for Computational Linguistics.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Scott Barnett, Stefanus Kurniawan, Srikanth Thudumu, Zach Brannelly, and Mohamed Abdelrazek. 2024. Seven failure points when engineering a retrieval augmented generation system. In *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering - Software Engineering for AI*, CAIN '24, page 194–199, New York, NY, USA. Association for Computing Machinery.

Sheng Bi, Xiya Cheng, Yuan-Fang Li, Lizhen Qu, Shirong Shen, Guilin Qi, Lu Pan, and Yinlin Jiang. 2021. Simple or complex? complexity-controllable question generation with soft templates and deep mixture of experts model. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4645–4654, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Shuyang Cao and Lu Wang. 2021. Controllable open-ended question generation with a new question type ontology. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6424–6439, Online. Association for Computational Linguistics.

Yllias Chali and Sadid A. Hasan. 2012. Towards automatic topical question generation. In *Proceedings of COLING 2012*, pages 475–492, Mumbai, India. The COLING 2012 Organizing Committee.

Yi Cheng, Siyao Li, Bang Liu, Ruihui Zhao, Sujian Li, Chenghua Lin, and Yefeng Zheng. 2021. Guiding the growth: Difficulty-controllable question generation through step-by-step rewriting. In *Proceedings*

of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 5968–5978, Online. Association for Computational Linguistics.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 866–874, Copenhagen, Denmark. Association for Computational Linguistics.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Zichu Fei, Qi Zhang, Tao Gui, Di Liang, Sirui Wang, Wei Wu, and Xuanjing Huang. 2022. CQG: A simple and effective controlled generation framework for multi-hop question generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6896–6906, Dublin, Ireland. Association for Computational Linguistics.

Zichu Fei, Qi Zhang, and Yaqian Zhou. 2021. Iterative GNN-based decoder for question generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2573–2582, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Shasha Guo, Lizi Liao, Cuiping Li, and Tat-Seng Chua. 2024. A survey on neural question generation: Methods, applications, and prospects. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 8038–8047. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

Michael Heilman. 2011. *Automatic factual question generation from text*. Carnegie Mellon University.

Seonjeong Hwang, Yunsu Kim, and Gary Geunbae Lee. 2024. Explainable multi-hop question generation: An end-to-end approach without intermediate question labeling. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 6855–6866, Torino, Italia. ELRA and ICCL.

Xin Jia, Wenjie Zhou, Xu Sun, and Yunfang Wu. 2021. Eqg-race: Examination-type question generation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 13143–13151.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Yanghoon Kim, Hwanhee Lee, Joongbo Shin, and Kyomin Jung. 2019. Improving neural question generation using answer separation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 6602–6609.

J. Kormos. 2006. *Speech Production and Second Language Acquisition*. Cognitive sciences and second language acquisition. Lawrence Erlbaum Associates.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461.

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.

Ziming Li, Youhuan Li, Yuyu Luo, Guoliang Li, and Chuxu Zhang. 2025. Graph neural networks for databases: A survey. *arXiv preprint arXiv:2502.12908*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. 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.

Yuxiang Liu, Jie Huang, and Kevin Chang. 2023. Ask to the point: Open-domain entity-centric question generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2703–2716, Singapore. Association for Computational Linguistics.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, and Adam Roberts. 2023. The flan collection: Designing data and methods for effective instruction tuning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 22631–22648. PMLR.

I Loshchilov. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Max Marion, Ahmet Üstün, Luiza Pozzobon, Alex Wang, Marzieh Fadaee, and Sara Hooker. 2023. When less is more: Investigating data pruning for pretraining llms at scale. *arXiv preprint arXiv:2309.04564*.

Jack Mostow and Wei Chen. 2009. Generating instruction automatically for the reading strategy of self-questioning. In *Artificial Intelligence in Education*, pages 465–472. IOS Press.

Lidiya Murakhovs'ka, Chien-Sheng Wu, Philippe Laban, Tong Niu, Wenhao Liu, and Caiming Xiong. 2022. Mixqg: Neural question generation with mixed answer types. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1486–1497.

Liangming Pan, Yuxi Xie, Yansong Feng, Tat-Seng Chua, and Min-Yen Kan. 2020. Semantic graphs for generating deep questions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1463–1475, Online. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. 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, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Megha Srivastava and Noah Goodman. 2021. Question generation for adaptive education. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 692–701, Online. Association for Computational Linguistics.

Dan Su, Peng Xu, and Pascale Fung. 2022. Qa4qg: using question answering to constrain multi-hop question generation. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8232–8236. IEEE.

Dan Su, Yan Xu, Wenliang Dai, Ziwei Ji, Tiezheng Yu, and Pascale Fung. 2020. Multi-hop question generation with graph convolutional network. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4636–4647, Online. Association for Computational Linguistics.

Xingwu Sun, Jing Liu, Yajuan Lyu, Wei He, Yanjun Ma, and Shi Wang. 2018. Answer-focused and position-aware neural question generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3930–3939, Brussels, Belgium. Association for Computational Linguistics.

Duyu Tang, Nan Duan, Tao Qin, Zhao Yan, and Ming Zhou. 2017. Question answering and question generation as dual tasks. *arXiv preprint arXiv:1706.02027*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. NewsQA: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200, Vancouver, Canada. Association for Computational Linguistics.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Masaki Uto, Yuto Tomikawa, and Ayaka Suzuki. 2023. Difficulty-controllable neural question generation for reading comprehension using item response theory. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 119–129, Toronto, Canada. Association for Computational Linguistics.

Bingning Wang, Xiaochuan Wang, Ting Tao, Qi Zhang, and Jingfang Xu. 2020. Neural question generation with answer pivot. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9138–9145.

Pei Wang, Keqing He, Yejie Wang, Xiaoshuai Song, Yutao Mou, Jingang Wang, Yunsen Xian, Xunliang Cai, and Weiran Xu. 2024. Beyond the known: Investigating LLMs performance on out-of-domain intent detection. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 2354–2364, Torino, Italia. ELRA and ICCL.

Zehua Xia, Qi Gou, Bowen Yu, Haiyang Yu, Fei Huang, Yongbin Li, and Nguyen Cam-Tu. 2023. Improving question generation with multi-level content planning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 800–814, Singapore. Association for Computational Linguistics.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Fuda Ye, Shuangyin Li, Yongqi Zhang, and Lei Chen. 2024. R^{2}AG: Incorporating retrieval information into retrieval augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11584–11596, Miami, Florida, USA. Association for Computational Linguistics.

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*.

Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. 2018. Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3901–3910, Brussels, Belgium. Association for Computational Linguistics.

Zhenjie Zhao, Yufang Hou, Dakuo Wang, Mo Yu, Chengzhong Liu, and Xiaojuan Ma. 2022. Educational question generation of children storybooks via question type distribution learning and event-centric summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5073–5085, Dublin, Ireland. Association for Computational Linguistics.

Tianqi Zhong, Zhaoyi Li, Quan Wang, Linqi Song, Ying Wei, Defu Lian, and Zhendong Mao. 2024. Benchmarking and improving compositional generalization of multi-aspect controllable text generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6486–6517, Bangkok, Thailand. Association for Computational Linguistics.

A Implementation Details

We utilize GPT-4o-mini-2024-07-18[4] from the OpenAI API for passage sorting and multi-hop skeleton construction. Additionally, we employ GPT-4o-2024-08-06[5] to decompose multi-hop questions from the original dataset to obtain pseudo (gold) sub-question-answer (QA) pairs for SQAG. To prevent overfitting during MHQG training, we apply simple noise injection to the sub-QA pairs in the MHQG training set. Specifically, we adopt a random sampling and rewriting approach to introduce noise into the data.

We use MixQG-base[6], a pretrained model specifically designed for question generation, with a total of 440M parameters, as our primary backbone model. Additionally, we report the performance of our framework on alternative backbone models, T5-base[7] and FLAN-T5-base[8] on MuSiQue, in App. D. FLAN-T5-base is a fine-tuned version of the T5 model, optimized for instruction-following tasks.

Seed 42 is used throughout the experiment. We adopt AdamW (Loshchilov, 2017) as the optimizer. All models are trained on a single NVIDIA A800-80G GPU with `bf16 = True` for RUBY and all foundational models, while other models use `bf16 = False`. Moreover, we set the batch size to 8, learning rate to 5e-5, warmup ratio to 0.1, epoch to 10, maximum source length to 512, and maximum target length to 128. For generation, we adopt greedy decoding across all models. Other parameters follow the default settings in the Huggingface Trainer and Generator configuration files.

B Experiments and Results on HotpotQA

We conduct comparative experiments against baselines as well as ablation studies on the proposed framework RUBY using HotpotQA. For the annotated sub-question-answer pairs used in SQAG, we leverage the high-quality sub-QA pairs available in MuSiQue to train a multi-hop question decomposition model based on T5-Large. This model is used to generate annotated sub-QA pairs for HotpotQA. Specifically, the input to this model is a multi-hop question along with a corresponding passage, and the output is the sub-question-answer pairs corresponding to that passage. The results of

[4]https://platform.openai.com/docs/models/gpt-4o-mini
[5]https://platform.openai.com/docs/models/gpt-4o
[6]https://huggingface.co/Salesforce/mixqg-base
[7]https://huggingface.co/google-t5/t5-base
[8]https://huggingface.co/google/flan-t5-base

Model	B-1	B-2	B-3	B-4	R-L	MTR	BSc
Llama-3-1-8B	60.50	46.62	37.86	31.55	50.02	32.27	60.46
Mistral-7B-v0.3	58.83	46.20	37.84	31.67	50.13	32.11	60.37
BART-base	58.47	46.85	39.04	33.28	50.33	32.90	60.15
T5-base	60.66	47.91	39.45	33.35	50.60	33.23	61.18
FLAN-T5-base	<u>61.43</u>	<u>48.12</u>	39.64	33.50	50.97	<u>33.53</u>	<u>61.21</u>
MixQG-base	59.78	47.45	39.53	33.73	<u>51.20</u>	31.72	60.77
CQG	23.12	10.58	5.20	2.80	21.25	11.58	11.36
E2EQR	59.25	45.42	36.63	30.38	47.80	31.16	57.92
MultiFactor	60.83	48.03	<u>39.81</u>	<u>33.84</u>	50.34	32.71	60.62
RUBY	**63.31**	**51.04**	**42.93**	**36.75**	**53.26**	**33.90**	**63.74**

Table 5: Automatic evaluation on HotpotQA. The **Bold** and <u>underline</u> mark the best and second-best results.

Model	B-1	B-2	B-3	B-4	R-L	BSc
RUBY	**63.31**	**51.04**	**42.93**	**36.75**	**53.26**	**63.74**
w/o RD-P	60.71 (↓ 4.11%)	48.55 (↓ 4.88%)	40.46 (↓ 5.75%)	34.40 (↓ 6.39%)	**53.26** (↓ 0.00%)	62.03 (↓ 2.68%)
w/o SQA	62.18 (↓ 1.78%)	50.26 (↓ 1.53%)	42.18 (↓ 1.75%)	36.14 (↓ 1.66%)	52.06 (↓ 2.25%)	61.66 (↓ 3.26%)
w/o HDR	60.90 (↓ 3.81%)	48.08 (↓ 5.80%)	39.90 (↓ 7.06%)	33.85 (↓ 7.89%)	51.77 (↓ 2.80%)	60.86 (↓ 4.52%)

Table 6: Ablation studies on HotpotQA showing BLEU, ROUGE-L, and BERTScore for RUBY.

comparative experiments against baselines are presented in Table 5, and the results of ablation studies on RUBY are presented in Table 6. From the results, it is evident that RUBY achieves the best performance on the HotpotQA dataset, demonstrating that RUBY is effective across multiple datasets. This highlights RUBY's strong generalization capability across different datasets.

C Complete Evaluation Results of Different-Hop Question Generation

We report the complete evaluation results of RUBY and various baseline models for multi-hop question generation on MuSiQue in Table 14.

RUBY consistently outperforms all baseline models across 2-hop, 3-hop, and 4-hop question generation tasks, demonstrating its strong reasoning and question formulation capabilities in multi-hop question generation. Notably, RUBY achieves superior performance compared to models such as T5-base and MultiFactor on key evaluation metrics, including BLEU, ROUGE-L, METEOR, and BERTScore, which assess the fluency, relevance, and overall quality of generated questions.

Furthermore, RUBY excels in maintaining logical coherence and adherence to constraints, as evidenced by its high scores in entity consistency, intent consistency, and multi-hop type consistency. These results highlight RUBY's ability to generate accurate and contextually consistent multi-hop questions across a variety of scenarios.

Model	C-1	C-2	C-3	Hop-Acc
Llama-3.1-8B	<u>0.7100</u>	0.9935	<u>0.6094</u>	<u>2.3129</u>
Mistral-7B-v0.3	0.6950	0.9771	0.5689	2.2410
BART-base	0.6061	0.9967	0.4912	2.0941
T5-base	0.6192	0.9967	0.5036	2.1195
FLAN-T5-base	0.6342	0.9980	0.5147	2.1470
MixQG-base	0.6695	<u>0.9993</u>	0.5526	2.2214
MulQG	0.1652	**1.0000**	0.1106	1.2758
CQG	0.2148	**1.0000**	0.1208	1.3356
E2EQR	0.6166	**1.0000**	0.4853	2.1019
MultiFactor	0.6558	0.9980	0.5402	2.1940
RUBY	**0.7172**	0.9980	**0.6218**	**2.3370**

Table 7: Comparison of models based on different criteria and Hop-Accuracy on MuSiQue. C-1, C-2, and C-3 represent Criterion-1, Criterion-2, and Criterion-3, respectively. Hop-Accuracy is the sum of all the scores of the criteria.

D RUBY's Generalization across Different Transformer Models

We conducted extensive experiments on MuSiQue to evaluate the performance of the RUBY framework across various Transformer architectures. As shown in Figure 6, RUBY consistently outperforms both T5-base and FLAN-T5-base across all evaluation metrics, highlighting its robust generalization ability. Specifically, RUBY-T5-base achieves higher scores in metrics including BLEU-4, ROUGE-L, METEOR, and BERTScore compared to T5-base. Similarly, RUBY-FLAN-T5-base outperforms FLAN-T5-base in these areas, while also showing superior performance in consistency metrics, including entity consistency, intent consistency, and multi-hop type consistency. The re-

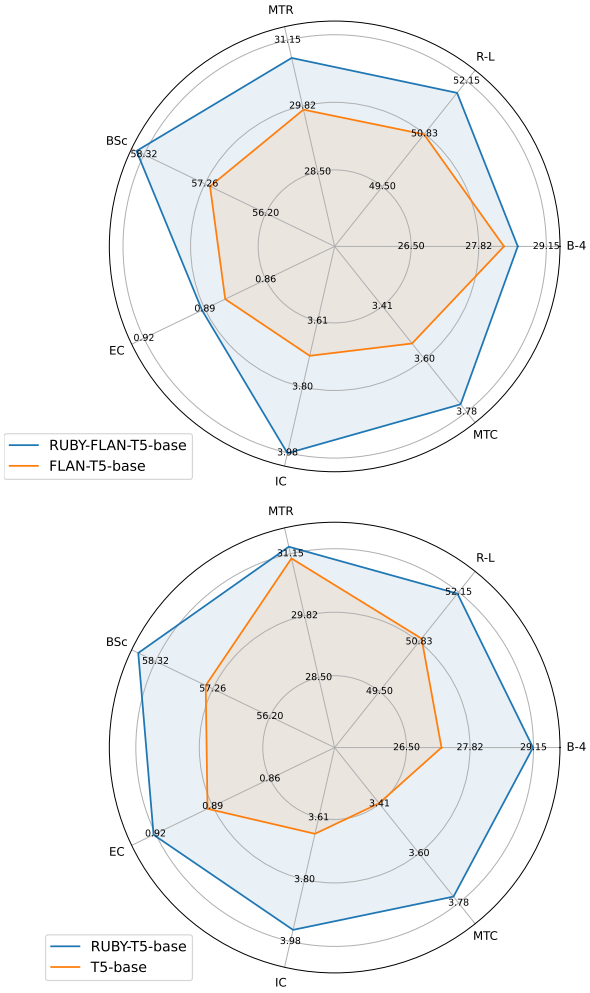


Figure 6: The results of RUBY on the backbone models T5-base and FLAN-T5-base on MuSiQue.

sults indicate that RUBY's architecture not only enhances the quality of generated questions but also improves the alignment with key semantic constraints, making it a highly effective approach across different Transformer backbone models.

E Evaluation of Constraint Consistency

For the entity constraint, since it involves comparison and judgment at the token level, we follow the following criteria to assess whether a given entity is contained in the generated question:

$$\text{Score}(e, q) = \begin{cases} 1 & \text{if } e \subseteq q, \\ 0 & \text{if } e \not\subseteq q, \end{cases} \quad (16)$$

where e represents the given entity and q represents the generated question. For evaluating the intent consistency and multi-hop type consistency constraints, we employ Llama-3.1-70B-Instruct[9] in combination with a specially designed prompt. Specifically, for intent consistency, when using the template, we replace the tokens [CONTEXT], [INTENT], [R_Q] and [Q] with the given context, intent, reference question, and generated question, respectively, and then input the entire prompt into Llama-3.1. Similarly, for multi-hop type consistency, when using the template, we replace the tokens [CONTEXT], [MT], [R_Q] and [Q] with the given context, multi-hop type, reference question, and generated question, respectively, and then input the entire prompt into Llama.

[9]https://huggingface.co/meta-llama/Llama-3.1-70B-Instruct

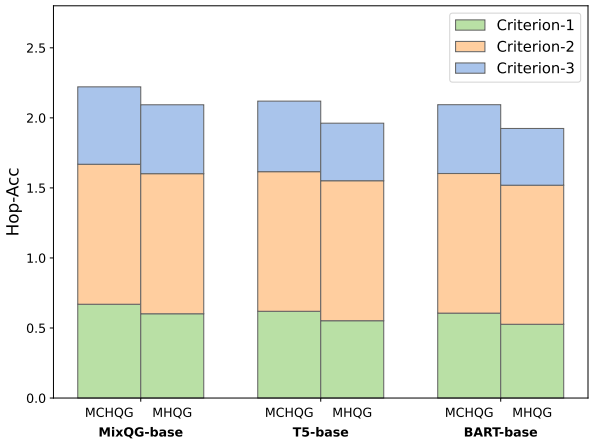


Figure 7: Hop-Accuracy (Hop-Acc) of MixQG-base, T5-base and BART-base on MCHQG task and MHQG task.

F Evaluation of Hop Error

We conducted an experiment on MixQG-base, T5-base, and BART-base for MHQG and MCHQG to evaluate the effectiveness of the multi-constraint setting in reducing multi-hop errors on MuSiQue. Based on the definition of multi-hop errors from Xia et al. (2023), we primarily evaluated the following three criteria: 1) whether the question content is fully derived from the passages, 2) whether the answer is present in the question, and 3) whether a reasoning chain leading to the answer can be inferred from the question content. For Criteria 1 and 3, we used the designed prompt shown in Table 17 with Llama-3.1-70B-Instruct to perform the evaluation. For Criterion 2, we used a rule-based approach: a score of 1 was assigned if the answer was not included in the question, and a score of 0 if it was present. Additionally, if Criterion 2 received a score of 0, Criterion 3 was automatically assigned a score of 0.

As shown in Figure 7, MixQG, T5, and BART achieve higher hop accuracy on MCHQG than on MHQG. This indicates that the multi-constraint setting in MCHQG reduces hop errors compared

Model	B-1	B-2	B-3	B-4	R-L	MTR	BSc
RUBY (full-input)	58.64	45.45	36.59	30.04	52.89	31.50	59.40
Simplified-Input (w/o entity)							
RUBY	**56.79**	**43.29**	**34.46**	**28.07**	**50.88**	**29.80**	**56.78**
FLAN-T5-base	55.14	41.20	32.32	25.93	48.13	26.97	53.21
MixQG-base	53.07	39.79	31.15	24.92	48.29	28.43	54.00
Simplified-Input (concise intent)							
RUBY	**56.17**	**43.04**	**34.30**	**27.93**	**51.19**	**30.94**	**57.88**
FLAN-T5-base	55.24	41.29	32.32	25.86	48.72	28.94	55.65
MixQG-base	52.31	39.69	31.29	25.16	49.80	30.61	56.68

Table 8: Performance of models with simplified input. The results marked with a gray background indicate that they are not directly comparable. "Full-input" refers to the input without any simplification. "w/o entity" refers to inputs that do not contain the specified entity. "Concise intent" refers to using more concise intents instead of the original ones.

to the answer-only constraint setting in MHQG.

Furthermore, as shown in Table 7, RUBY outperforms other models in Criterion-1, Criterion-3, and hop accuracy, demonstrating its effectiveness in reducing multi-hop errors. While its performance in Criterion-2 is very close to the best (1.0000), RUBY maintains a strong overall performance across all metrics, making it the most balanced model in the evaluation.

G Performance with Simplified Inputs

We explore the performance of RUBY when using simplified inputs during the inference phase. We set up two approaches for simplifying the inputs: using more concise intents instead of the original ones (concise intent) and omitting the specified entity (w/o entity). We conduct experiments on the MuSiQue dataset based on these two approaches for simplifying the inputs. For the first approach, we use GPT-4o-mini to concisely rewrite the intents in MuSiQue's test set, resulting in an average intent length of 8.21, a 41.61% reduction compared to the original length of 14.06.

The experimental results, as shown in Table 8, demonstrate that even with simplified inputs, RUBY achieves commendable performance and surpasses most of the baselines using full inputs (without simplification), as presented in Table 1, except for FLAN-T5-base and MixQG-base. We further conduct experiments where FLAN-T5-base and MixQG-base also used the simplified-input setup. The results, also presented in Table 8, show that, under the same input settings, RUBY still outperforms both FLAN-T5-base and MixQG-base.

H Generalization Performance of RUBY without Annotated Sub-QA Pairs

The training and inference phases are separate. Since MuSiQue provides high-quality sub-QA pairs, a model trained on MuSiQue can generalize effectively to other datasets. Specifically, the models used for SQAG and MHQG on other datasets are pre-trained on MuSiQue, meaning that no sub-QA pair annotations are required for these datasets.

Table 9 presents the performance of the models pre-trained on MuSiQue when applied to the HotpotQA test set. As shown, RUBY demonstrates superior generalization capability compared to other models. The results indicate that RUBY, despite not relying on annotated sub-QA pairs in the target dataset, achieves the highest performance across several metrics. This highlights RUBY's strong ability to generalize across datasets while maintaining competitive performance without the need for dataset-specific annotations.

I Passage Sorting Algorithm

Algorithm 1 describes the full implementation of the process of passage sorting.

J Training Algorithm

Algorithm 2 describes the full implementation of the process of the training for SQAG and MHQG.

K Data Processing Workflow

To impose additional constraints for the MCHQG task, we extract intent, entity, and multi-hop type information.

Model	B-1	B-2	B-3	B-4	R-L	MTR	BSc
Llama-3.1-8B	48.65	33.18	23.91	17.83	35.31	26.63	**49.18**
Mistral-7B-v0.3	46.86	32.19	23.45	17.59	35.50	24.66	46.95
FLAN-T5-base	47.94	32.46	23.26	17.11	34.37	25.18	45.91
MixQG-base	48.07	33.16	24.07	18.05	**35.75**	27.25	48.49
MultiFactor	43.21	28.42	19.96	14.54	31.20	22.99	41.89
RUBY	**49.94**	**34.46**	**24.90**	**18.54**	35.66	**27.60**	48.89

Table 9: Performance comparison of models pre-trained on MuSiQue and applied to the HotpotQA test set.

Algorithm 1: Passage Sorting

Input : Passages $\mathcal{P}$, answer a, multi-hop type $\mathcal{M}$, intent it, the number of passages ℓ.

Output : Sorted sequence of passages $seq^{\mathcal{P}}$, sequence excluding the answer $seq^{\mathcal{P}}_{\setminus a}$.

Initialize the initial selected passages $\mathcal{P}^s_0 = \{a\}$, initial candidate passages $\mathcal{P}^c_0 = \mathcal{P}$.

for $i = 1$ **to** ℓ **do**

- $\mathcal{P}' = \mathcal{P} \cup \{a\}$;
- Divide $\mathcal{P}'$ into $\mathcal{P}^c_i$ and $\mathcal{P}^s_i$;
- $seq_{i-1} = f_{ig}(\mathcal{M}, \mathcal{P}^s_{i-1})$;
- $p^*_i = \text{LLM}(T^{\text{sort}}(it, seq_{i-1}, \mathcal{P}^c_{i-1}))$;
- $\mathcal{P}^s_i = \mathcal{P}^s_{i-1} \cup \{p^*_i\}$;
- $\mathcal{P}^c_i = \mathcal{P}' \setminus \mathcal{P}^s_i$;
- Ensure $\mathcal{P}^c_i \cap \mathcal{P}^s_i = \emptyset$.

end

$seq^{\mathcal{P}} = seq_\ell = f_{ig}(\mathcal{M}, \mathcal{P}^s_\ell)$.

$seq^{\mathcal{P}}_{\setminus a} = seq^{\mathcal{P}} \setminus [a]$.

return $seq^{\mathcal{P}}$, $seq^{\mathcal{P}}_{\setminus a}$

For MuSiQue, we use GPT-4o-2024-08-06 via the OpenAI API to extract both the intent and entities. Then, we randomly select one entity that includes an uppercase letter as the final choice, ensuring reproducibility using a seed value of 42. To construct the multi-hop type, we utilize the "question_decomposition" field from MuSiQue, with the resulting multi-hop types summarized in Table 15. Supporting paragraphs from the dataset are selected as passages. We filter out examples that lack an entity with an uppercase letter. After filtering, the final dataset consists of 13,459/823/1,531 examples for training, validation, and testing on MuSiQue. The dataset statistics are shown in Table 11.

For HotpotQA, we use seed 42 to randomly select 1,000 examples from the training set, 200 from the validation set, and another 200 from the original validation set to create new training, validation, and test sets. We utilize GPT-4o to assign a multi-hop type to each example according to the reasoning type definitions outlined in the original HotpotQA paper. Since the multi-hop type represents the logical structure of a question, we subsequently recruit 10 professional volunteers to manually correct any incorrect multi-hop type annotations. The results of the multi-hop type annotations are summarized in Table 15. For intents and entities, we adopt the same construction methodology as used for the MuSiQue dataset. We select the supporting paragraphs as the passages (not supporting facts). The dataset statistics are shown in Table 12.

To demonstrate that the quality of intents and entities generated using LLMs is reliable, we conducted experiments to evaluate the quality of the generated intents. We employ Perplexity (PPL), a commonly used quantitative metric for assessing data quality (Marion et al., 2023; Zhong et al., 2024). Since intents are primarily generated based on the questions in the original dataset, we compared the PPL of the intents with that of the corresponding questions.

Here, PPL is computed using GPT-2. As shown in Table 10, the PPL of intents is significantly lower than that of the questions across all datasets, indicating that the quality of the intents generated by GPT-4o is reasonably ensured.

Furthermore, existing studies have demonstrated that LLMs perform remarkably well on intent detection tasks in zero-shot/few-shot settings (Arora et al., 2024; Wang et al., 2024), further ensuring the quality of the generated intents. As for the entities, we only retained those extracted by the LLMs that also appear in the original question, thereby guaranteeing their reliability.

Algorithm 2: Training

Input : Training datasets $\mathcal{D}_s$ and $\mathcal{D}_m$.

Output : Trained model parameters ξ_s and ξ_m.

Stage 1: Training SQAG

Initialize the model parameters ξ_s.

for $k_s = 1$ **to** $|\mathcal{D}_s|$ **do**

  Get input $x^{(k_s)}$ and output $y^{(k_s)}$;

  Compute the loss:

$$\mathcal{L}_s = -\sum_{l_s=1}^{|y^{(k_s)}|} \log P_s\left(y_{l_s}^{(k_s)} \middle| x^{(k_s)}; \xi_s\right);$$

  Update ξ_s via gradient descent on $\mathcal{L}_s$.

end

Stage 2: Training MHQG

Initialize the model parameters ξ_m.

for $k_m = 1$ **to** $|\mathcal{D}_m|$ **do**

  Get input $x^{(k_m)}$ and output $y^{(k_m)}$;

  Compute the loss:

$$\mathcal{L}_m = -\sum_{l_m=1}^{|y^{(k_m)}|} \log P_m\left(y_{l_m}^{(k_m)} \middle| x^{(k_m)}; \xi_m\right);$$

  Update ξ_m via gradient descent on $\mathcal{L}_m$.

end

return: ξ_s and ξ_m

L Effectiveness of Questions in Question Answering Task

To evaluate the effectiveness of different models, we apply the generated multi-hop questions to downstream QA tasks. Following the evaluation protocols of Barnett, Barnett et al. (2024), and Ye et al. (2024), we measure the performance of QA systems on the MuSiQue and HotpotQA datasets using accuracy (Acc) and F1 score. We employ Llama 3.1-70B as the QA model to answer questions generated by FLAN-T5-base, MultiFactor, Llama3.1-8B-Instruct, and RUBY. As shown in Table 13, questions generated by RUBY yield higher answerability and are more semantically aligned with the golden questions than those generated by other representative baselines.

Dataset	Type	Intent PPL (↓)	Question PPL (↓)
MuSiQue	Training	78.31	218.09
	Validation	79.15	214.16
	Test	78.63	217.54
HotpotQA	Training	73.38	293.67
	Validation	81.25	324.66
	Test	76.76	268.17

Table 10: Evaluation of the quality of the generated intents using Perplexity (PPL).

Metric	Training	Validation	Test
Passages Mean	198.48	239.65	243.14
Passages Min	46	47	45
Passages Max	805	645	683
Question Mean	15.93	17.84	18.40
Question Min	4	6	5
Question Max	46	46	43
Answer Mean	2.37	2.40	2.88
Answer Min	1	1	1
Answer Max	14	14	14
Intent Mean	13.72	16.05	14.06
Intent Min	6	7	6
Intent Max	34	28	29
Entity Mean	2.01	1.93	2.19
Entity Min	1	1	1
Entity Max	11	8	12
2-hop type	9845	410	764
3-hop-1 type	2310	199	371
3-hop-2 type	496	69	128
4-hop-1 type	440	72	133
4-hop-2 type	71	28	51
4-hop-3 type	297	45	84

Table 11: The statistic of MuSiQue. The mean, min, max lengths of passages, questions, answers, intents, and entities are reported. The number of each multi-hop type is also reported.

M Terminology and Explanation

Table 16 shows the key terminologies and their explanations.

N A Case Study of RUBY

Figure 8 presents a case study of RUBY, illustrating the detailed process of its high-dimensional semantic constraint dimension reduction module and sub-QA-based multi-hop question generation module.

O Prompt Templates used in HDR

Table 18 shows the prompt template used in the process of passage sorting. In each iteration, we select one passage from the candidate passages based on the other given information to correspond to #[NUM] in the multi-hop type. Specifically, we replace the tokens [SELECTED_PASSAGES], [CANDIDATE_PASSAGES], [NUM], [ANSWER], and [INTENT] with the corresponding information.

Table 19 shows the prompt template used in the construction of the multi-hop skeleton. Specifically, we replace the tokens [ENTITY], [ANSWER], [INTENT], and [SEQ] with the given entity, answer,

Metric	Training	Validation	Test
Passages Mean	145.66	137.93	141.24
Passages Min	31	45	35
Passages Max	345	314	295
Question Mean	15.17	15.30	15.27
Question Min	6	7	6
Question Max	38	38	30
Answer Mean	2.10	2.00	2.16
Answer Min	1	1	1
Answer Max	14	9	8
Intent Mean	14.81	14.61	14.69
Intent Min	7	7	7
Intent Max	27	24	25
Entity Mean	2.52	2.45	2.37
Entity Min	1	1	1
Entity Max	14	7	7
2-hop type	352	60	76
2-hop-bridge type	369	74	67
2-hop-comparison type	279	66	57

Table 12: The statistic of HotpotQA. The mean, min, max lengths of passages, questions, answers, intents, and entities are reported. The number of each multi-hop type is also reported.

Model	MuSiQue		HotpotQA	
	F1	Acc	F1	Acc
FLAN-T5-base	0.3859	0.4206	0.7329	0.7450
MultiFactor	0.3752	0.4115	0.7549	0.7600
Llama3.1-8B	0.4126	0.4448	0.7197	0.7350
RUBY	**0.4323**	**0.4513**	**0.7945**	**0.8100**
Golden Labels	0.4655	0.4827	0.8001	0.8250

Table 13: The effectiveness of questions generated by different methods in question answering tasks on the MuSiQue and HotpotQA datasets. Numbers in **Bold** indicate the best model performance, excluding those obtained using golden labels. The results marked with a gray background indicate that they are not directly comparable.

intent, and the sorted sequence of passages. Additionally, we replace the token [EXAMPLE] with an example that includes a Chain-of-Thought consisting of information analysis, extraction, and multi-hop skeleton construction.

Model	B-1	B-2	B-3	B-4	R-L	MTR	BSc
			2-hop				
Llama-3.1-8B	63.37	50.19	41.43	34.70	55.55	33.85	64.93
Mistral-7B-v0.3	61.82	48.82	40.21	33.74	54.29	32.98	64.31
BART-base	62.91	50.53	42.22	35.82	57.37	35.54	66.06
T5-base	64.32	51.87	43.40	36.79	57.93	35.07	66.16
FLAN-T5-base	64.12	51.66	43.03	36.42	57.56	34.69	65.89
MulQG	49.11	33.66	24.92	18.90	40.44	23.63	37.83
GenCONE	41.79	27.32	18.99	13.60	34.72	19.47	44.64
CQG	43.13	29.59	21.81	16.41	42.89	26.09	40.37
E2EQR	60.34	47.73	39.39	33.03	55.39	34.18	63.92
MultiFactor	63.54	50.84	42.19	35.49	55.80	34.29	64.74
MixQG-base	64.35	51.95	43.46	36.84	58.21	35.41	66.38
RUBY	**66.50**	**54.00**	**45.36**	**38.61**	**59.10**	**35.82**	**67.52**
			3-hop				
Llama-3.1-8B	52.16	37.91	28.74	22.43	44.41	27.87	50.23
Mistral-7B-v0.3	51.57	37.59	28.82	22.74	43.75	27.48	50.07
BART-base	51.85	38.98	30.35	24.09	47.24	30.20	52.64
T5-base	51.45	38.37	29.50	23.07	46.09	29.44	51.74
FLAN-T5-base	54.76	40.67	31.57	25.17	46.14	28.08	51.25
MulQG	27.69	14.14	7.49	4.07	22.47	13.11	12.17
GenCONE	34.62	22.69	15.79	11.40	31.44	20.25	37.67
CQG	32.21	20.76	14.09	9.74	34.89	22.35	30.96
E2EQR	52.81	40.32	32.01	25.94	48.71	**31.06**	53.98
MultiFactor	50.98	37.67	29.00	22.91	45.39	29.03	51.13
MixQG-base	51.41	38.64	29.97	23.77	47.27	29.88	52.52
RUBY	**55.99**	**42.60**	**33.54**	**27.03**	**49.17**	30.76	**54.42**
			4-hop				
Llama-3.1-8B	47.41	32.94	24.05	18.14	37.45	24.16	41.90
Mistral-7B-v0.3	46.27	33.00	24.66	18.79	38.99	24.40	42.44
BART-base	46.10	32.96	24.43	18.33	39.27	**26.45**	44.59
T5-base	44.95	32.16	23.91	18.23	40.21	26.01	43.57
FLAN-T5-base	49.30	35.57	26.87	20.78	40.24	24.76	43.33
MulQG	25.08	12.83	7.09	3.88	21.55	13.44	11.75
GenCONE	26.67	17.20	12.03	8.89	29.88	20.79	34.42
CQG	27.20	16.56	10.93	7.34	29.89	19.55	24.00
E2EQR	43.80	31.22	23.55	18.13	40.01	24.58	42.70
MultiFactor	46.91	33.42	25.12	19.49	39.99	26.19	43.72
MixQG-base	45.72	32.42	24.31	18.65	40.68	25.70	44.46
RUBY	**49.97**	**36.15**	**27.34**	**21.20**	**42.09**	25.78	**45.52**

Table 14: Comparison of RUBY and baseline models for 2-hop, 3-hop, and 4-hop configurations on MuSiQue. The best results in each category are highlighted in **bold**. The suboptimal results in each category are underlined.

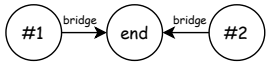
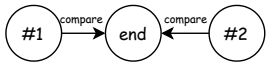
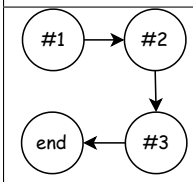
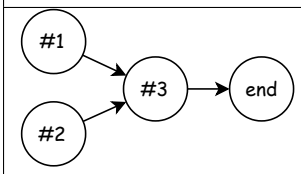
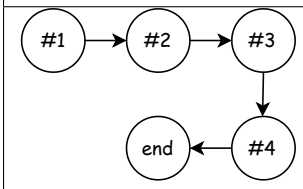
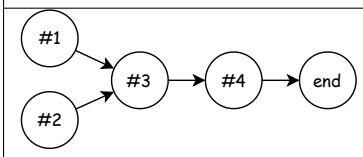
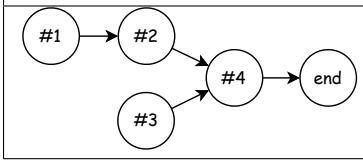
Graph Representation	Example	Description
	Q: When was the institute that owned The Collegian founded? A: 1960	This multi-hop type is #1 -> #2 -> end, which means a 2-hop type. This type exists in both MuSiQue and HotpotQA. The description in HotpotQA is "inferring the bridge entity to complete the second-hop question or inferring the property of an entity in question through a bridge entity."
	Q: Which former member of the Pittsburgh Pirates was nicknamed "The Cobra"? A: Dave Parker	This multi-hop type is #1 -> end <- #2, which means a 2-hop-bridge type. The description in HotpotQA is "locating the answer entity by checking multiple properties."
	Q: Did LostAlone and Guster have the same number of members? A: yes	This multi-hop type is #1 => end <= #2, which means a 2-hop-comparison type. The description in HotpotQA is "comparing two entities."
	Q: When did public health start in the country where the organization that recognizes the sovereignty of the Republic of Cyprus is headquartered? A: 1798	This multi-hop type is #1 -> #2 -> #3 -> end, which means a 3-hop-1 type.
	Q: Who started the Bethel branch of the religion founded by the black community in the city that used to be the US capitol? A: Bishop Francis Asbury	This multi-hop type is #1 -> #3 and #2 -> #3 and #3 -> end, which means a 3-hop-2 type.
	Q: Where is the lowest place in the country which, along with Eisenhower's VP's country, recognized Gaddafi's government early on? A: Holme Fen, Cambridgeshire	This multi-hop type is #1 -> #2 -> #3 -> #4 -> end, which means a 4-hop-1 type.
	Q: How many square miles is the source of the most legal immigrants to the location of Gotham's filming from the region where Andy from The Office sailed to? A: 18,705	This multi-hop type is #1 -> #3 and #2 -> #3 and #3 -> #4 -> end, which means a 4-hop-2 type.
	Q: What nation provided the most legal immigrants to the city that Gotham is filmed in, in the area containing the island with Philipsburg as its capital? A:the Dominican Republic	This multi-hop type is #1 -> #2 -> #4 and #3 -> #4 and #4 -> end, which means a 4-hop-3 type.

Table 15: The eight multi-hop types (ranging from 2-hop to 4-hop) are derived from the graph representations, example QA pairs, and descriptions in Trivedi et al. (2022), as well as the types of multi-hop reasoning, example QA pairs, and descriptions in Yang et al. (2018). Text colors indicate different nodes: teal for #1, blue for #2, orange for #3, purple for #4, and red for end.

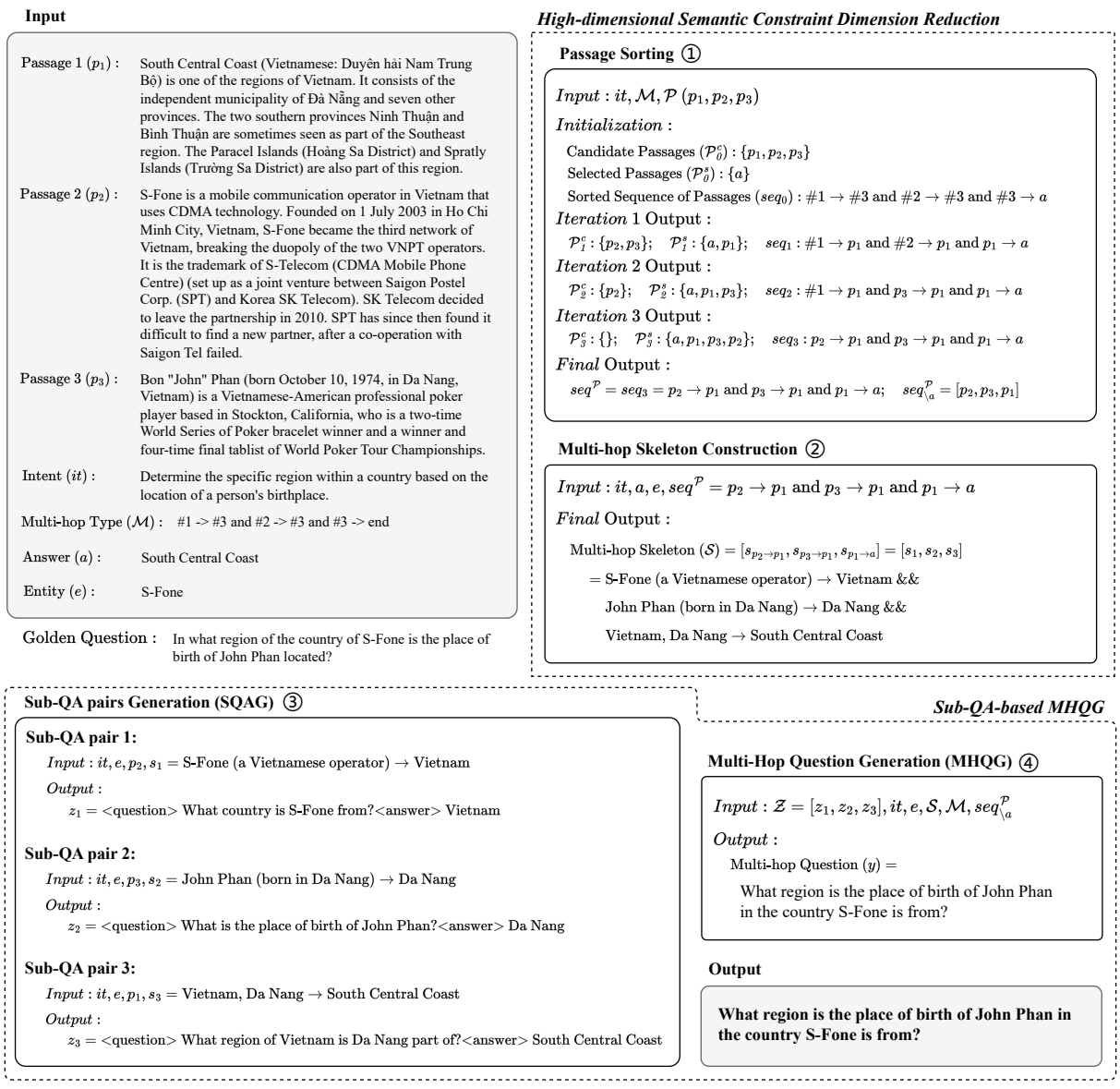


Figure 8: A case study of RUBY. ① to ④ shows the steps.

Terminology	Explanation
MHQG	Traditional Multi-Hop Question Generation.
MCHQG	Multi-Constraint Multi-Hop Question Generation, which is a **new task** we present.
Directionality	The preverbal plan before human questioning provides directionality for question generation, ensuring that the generated question is unique and precise rather than random.
High-dimensional and low-dimensional semantic constraint	A semantic constraint can be any word or sentence. In the MCHQG task, semantic constraints including entity, intent, and multi-hop type determine the direction of question generation. • Entity is considered a low-dimensional semantic constraint because the only requirement is that the generated question contains the given entity. • Intent and multi-hop type are high-dimensional semantic constraints because, unlike entities, they require the generated question to semantically align with them rather than simply include them as words.
High-dimensional Semantic Constraint Dimension Reduction (HDR)	A module for reducing high-dimensional semantic constraints, consisting of two stages: Passage Sorting and Multi-hop Skeleton Construction.
Passage Sorting	An iterative passage sorting process using LLMs.
Multi-hop Skeleton Construction	Generating multi-hop skeletons using LLMs.
Sub-Question Answer pair-based Multi-Hop Question Generation (SQA-based MHQG)	A module that decomposes multi-hop question generation into the generation of multiple sub-question-answer pairs, which are then synthesized into a single multi-hop question. This module incorporates Reasoning Dynamic Projection.
Reasoning Dynamic Projection (RD-Projection; RD-P)	An enhanced Transformer decoder that integrates a Block-wise State Enhancer.
Block-wise State Enhancer (BSE)	Enhancing representations in a block-wise manner. It consists of two components: Weight Calculator and Weight Broadcaster.
Weight Calculator and Weight Broadcaster	The Weight Calculator calculates the weight for each constraint, while the Weight Broadcaster broadcast the calculated weights to the corresponding representations of each constraint.

Table 16: Terminology and Explanation.

Prompt Template of Hop Error Evaluation

The following are several passages, a multi-hop question generated from them and the question's answer. Rate the following multi-hop question following the below criteria:
Criterion_1: If the key content of the question is fully derived from the passages, give a score of 1, otherwise give 0.
Criterion_2: If there is a complete reasoning chain to answer the question from the passages, give a score of 1, otherwise give 0.
Criterion_3: If there is a reasoning chain leading from the question to the answer that can be inferred from the question and the passage, give a score of 1, otherwise give 0. The scores should be in JSON format, for example:

```
{
    "Criterion_1": your_score for criterion_1 (0 or 1),
    "Criterion_2": your_score for criterion_2 (0 or 1),
    "Criterion_3": your_score for criterion_3 (0 or 1)
}
```

Passages: [CONTEXT]
Answer: [ANSWER]
Here is a reference version:
Question: [R_Q]
Scores:

```
{
    "Criterion_1": 1,
    "Criterion_2": 1,
    "Criterion_3": 1
}
```

The following question is the one you need to rate:
Question: [Q]
Scores:
<|eot_id|>

Table 17: Prompt template used for hop error evaluation. For both two criteria, a score of 1 represents the best result. In the template, [CONTEXT], [ANSWER], [R_Q] and [Q] are special tokens and need to be substituted.

Prompt Template of the Process of Passage Sorting

Task:
Identify the paragraph that logically precedes and connects to the known sequence in a multi-hop question setup. Your selection should be based on any relevant information that links the candidate paragraph to the already selected sequence. Noted that this time you should replace the #[NUM] in the Existing Sequence.
Instructions:
1. Review the given answer and question intent to understand the overarching theme.
2. Assess the existing sequence.
- For example, 'A -> B' in existing sequence means A shares some relevant key entities with B.
3. Please extract the key entities from all the paragraphs beforehand, and then, from the candidate paragraphs, choose one that shares any relevant key entities with the existing sequence.
4. Replace #{ num } to the number of the paragraph you select, reflecting its logical placement before the current content.

Existing Sequence: [SELECTED_PASSAGES]
Candidate Passages: [CANDIDATE_PASSAGES]
Output Required:
Provide the updated sequence, replacing #[NUM] with the selected paragraph that fits logically before the existing sequence.
For example, if you select paragraph 2, you should return:

```
{
    "#[NUM]": 2
}
```

Answer: [ANSWER]
Intent: [INTENT]
Output:

Table 18: Prompt template used for passage sorting. In the template, [SELECTED_PASSAGES], [CANDIDATE_PASSAGES], [NUM], [ANSWER] and [INTENT] are special tokens and need to be substituted.

Prompt Template for the Construction of Multi-Hop Skeleton

Task:
Given an entity, an answer, an intent, and sequences of paragraphs, please analyze how the paragraphs are related to each other and finally build the skeletons for the sequences. Focus on why such sequences are constructed, keeping the relationships between them clear and relevant to the entity, intent, and answer.

Instructions:
1. Do not introduce irrelevant information or redundant statements.
2. Do not hallucinate or assume details that are not provided in the context.
3. Analyze the connections between paragraphs based on the entity, answer, and intent, paying attention to how they logically follow from one another.
4. Provide a clear step-by-step analysis of how the paragraphs connect and why they are constructed this way, focusing on their relevance to the question's context.
5. When a "->" appears between two paragraphs, explain the relationship between them: This indicates that the second paragraph expands on, clarifies, or builds upon the context presented in the first one.
6. After analyzing, build the multi-hop skeletons using the key information from the paragraphs in the following JSON format:

```
{
    "Sequence 1": "...",
    "Sequence 2": "...",
    ......
}
```

[EXAMPLE]

Entity: [ENTITY]
Answer: [ANSWER]
Intent: [INTENT]
Sequences of paragraphs: [SEQ]
Output:

Table 19: Prompt template used for the construction of multi-hop skeleton. The special tokens [ENTITY], [ANSWER], [INTENT], and [SEQ] need to be substituted with actual content.

Prompt Template of Intent Consistency Evaluation

The following are a passage and a multi-hop question generated from it. Rate the following multi-hop question based on the corresponding passage and intent with respect to match degree, using a score from 1 to 5, where score 1 means "mismatch" and score 5 means "perfect match". Note that the match degree measures how well the question matches the intent, as well as the degree to which the nonspecific information and relationships in the intent are concretized from the passages. Only provide the score.
The score should be in JSON format, for example:

```
{
    "score": your_score (int)
}
```

Passage: [CONTEXT]
Intent: [INTENT]
Here is a reference version:
Question: [R_Q]
Score:

```
{
    "score": 5
}
```

Question: [Q]
Score:
<|eot_id|>

Table 20: Prompt template used for consistency evaluation on the intent. In the template, [CONTEXT], [INTENT], [R_Q] and [Q] are special tokens and need to be substituted.

Prompt Template of Multi-hop Type Consistency Evaluation

The following are passages and a multi-hop question generated from it. Rate the following multi-hop question based on the corresponding passages and multi-hop type with respect to match degree, using a score from 1 to 5, where score 1 means "mismatch" and score 5 means "perfect match". Note that match degree measures how well the question matches the multi-hop type. Only provide the score.
An explanation for the multi-hop type: each #number corresponds to some key information of one passage, if there is a -> between two #number, this means there are some relationships between the key information of two passages.
The score should be in JSON format, for example:

```
{
    "score": your_score (int)
}
```

Passages: [CONTEXT]
Multi-hop Type: [MT]
Here is a reference version:
Question: [R_Q]
Score:

```
{
    "score": 5
}
```

Question: [Q]
Score:
<leot_idl>

Table 21: Prompt template used for consistency evaluation on the multi-hop type. In the template, [CONTEXT], [MT], [R_Q] and [Q] are special tokens and need to be substituted.