White Rose Research Online

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk
Universities of Leeds, Sheffield and York

Deposited via The University of Sheffield.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/id/eprint/177026/

Version: Published Version

Proceedings Paper:
Cheng, Y., Li, S., Liu, B. et al. (2021) Guiding the growth : difficulty-controllable question generation through step-by-step rewriting. In: Zong, C., Xia, F. and Navigli, R., (eds.) Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing. 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2021), 01-06 Aug 2021, Bangkok, Thailand (virtual conference). Association for Computational Linguistics (ACL), pp. 5968-5978. ISBN: 9781954085527.

https://doi.org/10.18653/v1/2021.acl-long.465

© 2021 Association for Computational Linguistics. Licensed on a Creative Commons Attribution 4.0 International License (https://creativecommons.org/licenses/by/4.0/).

Reuse
This article is distributed under the terms of the Creative Commons Attribution (CC BY) licence. This licence allows you to distribute, remix, tweak, and build upon the work, even commercially, as long as you credit the authors for the original work. More information and the full terms of the licence here: https://creativecommons.org/licenses/

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

Guiding the Growth: Difficulty-Controllable Question Generation through Step-by-Step Rewriting

Yi Cheng[1], **Siyao Li**[2], **Bang Liu**[3]*, **Ruihui Zhao**[1], **Sujian Li**[4], **Chenghua Lin**[5], **Yefeng Zheng**[1]
[1]Tencent Jarvis Lab, China [2]LTI, Carnegie Mellon University
[3]RALI & Mila, Université de Montréal [4]Peking University [5]The University of Sheffield
{yicheng, zacharyzhao, yefengzheng}@tencent.com,
siyaol@andrew.cmu.edu, bang.liu@umontreal.ca,
lisujian@pku.edu.cn, c.lin@shef.ac.uk

Abstract

This paper explores the task of Difficulty-Controllable Question Generation (DCQG), which aims at generating questions with required difficulty levels. Previous research on this task mainly defines the difficulty of a question as whether it can be correctly answered by a Question Answering (QA) system, lacking interpretability and controllability. In our work, we redefine question difficulty as the number of inference steps required to answer it and argue that Question Generation (QG) systems should have stronger control over the logic of generated questions. To this end, we propose a novel framework that progressively increases question difficulty through step-by-step rewriting under the guidance of an extracted reasoning chain. A dataset is automatically constructed to facilitate the research, on which extensive experiments are conducted to test the performance of our method.

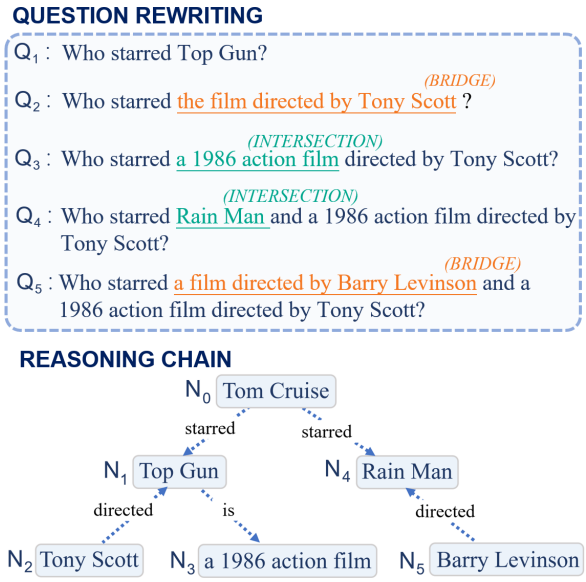


Figure 1: An example of generating a complex question through step-by-step rewriting based on the reasoning chain extracted from a constructed context graph.

1 Introduction

The task of Difficulty-Controllable Question Generation (DCQG) aims at generating questions with required difficulty levels and has recently attracted researchers' attention due to its wide application, such as facilitating certain curriculum-learning-based methods for QA systems (Sachan and Xing, 2016) and designing exams of various difficulty levels for educational purpose (Kurdi et al., 2020).

Compared to previous QG works which control the interrogative word (Zi et al., 2019; Kang et al., 2019) or the context of a question (Liu et al., 2020, 2019a), few works have been conducted on difficulty control, as it is hard to formally define the difficulty of a question. To the best of our knowledge, Gao et al. (2019) is the only previous work of DCQG for free text, and defines question difficulty as whether a QA model can correctly answer it. This definition gives only two difficulty levels and is mainly empirically driven, lacking interpretability for what difficulty is and how difficulty varies.

In this work, we redefine the difficulty level of a question as *the number of inference steps required to answer it*, which reflects the requirements on reasoning and cognitive abilities (Pan et al., 2019). Existing QA systems perform substantially worse in answering multi-hop questions than single-hop ones (Yang et al., 2018), also supporting the soundness of using reasoning hops to define difficulty.

To achieve DCQG with the above definition, a QG model should have strong control over the logic and reasoning complexity of generated questions. Graph-based methods are well suited for such logic modelling (Pearl and Paz, 1986; Zhang et al., 2020). In previous QG researches, Yu et al. (2020) and Pan et al. (2020) implemented graph-to-sequence

*Corresponding author.

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 5968–5978
August 1–6, 2021. ©2021 Association for Computational Linguistics

frameworks to distill the inner structure of the context, but they mainly used graphs to enhance document representations, rather than to control the reasoning complexity of questions.

In this paper, we propose a highly-controllable QG framework that progressively increases difficulties of the generated questions through step-by-step rewriting. Specifically, we first transform a given raw text into a context graph, from which we sample the answer and the reasoning chain for the generated question. Then, we design a question generator and a question rewriter to generate an initial simple question and step-by-step rewrite it into more complex ones. As shown in Fig. 1, "Tom Cruise" is the selected answer, and Q_1 is the initial question, which is then adapted into Q_2 by adding one more inference step (i.e. $N_1 \leftarrow N_2$) in the reasoning chain. That is, it requires to infer "Top Gun" is "the film directed by Tony Scott" before answering Q_1. Similarly, we can further increase its difficulty level and step-by-step extend it into more difficult questions (i.e., Q_3, Q_4 and Q_5).

To train our DCQG framework, we design effective strategies to automatically construct the training data from existing QA datasets instead of building one from scratch with intensive human efforts. Specifically, we utilize HotpotQA (Yang et al., 2018), a QA dataset where most questions require two inference steps to answer and can be decomposed into two 1-hop questions. Thus, we get the dataset that contains 2-hop questions and their corresponding 1-hop reasoning steps. Having learned how to rewrite 1-hop questions into 2-hop ones with this dataset, our framework can easily extend to the generation of (n+1)-hop questions from n-hop ones only with a small amount of corresponding data, because the rewriting operation follows rather certain patterns regardless of the exact value of n, as shown in Fig. 1.

Extensive evaluations show that our method can controllably generate questions with required difficulty, and keep competitive question quality at the same time, compared with a set of strong baselines.

In summary, our contributions are as follows:

- To the best of our knowledge, this is the first work of difficulty-controllable question generation, with question difficulty defined as the inference steps to answer it;
- We propose a novel framework that achieves DCQG through step-by-step rewriting under the guidance of an extracted reasoning chain;
- We build a dataset that can facilitate training of rewriting questions into more complex ones, paired with constructed context graphs and the underlying reasoning chain of the question.

2 Related Work

Deep Question Generation Most of the previous QG researches (Zhou et al., 2017; Pan et al., 2019; Liu et al., 2020) mainly focused on generating single-hop questions like the ones in SQuAD (Rajpurkar et al., 2016). In the hope that AI systems could provoke more in-depth interaction with humans, deep question generation aims at generating questions that require deep reasoning. Many recent works attempted to conquer this task with graph-based neural architectures. Talmor and Berant (2018) and Kumar et al. (2019) generated complex questions based on knowledge graphs, but their methods could not be directly applied to QG for free text, which lacks clear logical structures. In sequential question generation, Chai and Wan (2020) used a dual-graph interaction to better capture context dependency. However, they considered all the tokens as nodes, which led to a very complex graph. Yu et al. (2020) tried to generate multi-hop questions from free text with the help of entity graphs constructed by external tools. Our work shares a similar setting with Yu et al. (2020), and we further explore the problem of how to generate deep questions in a more controllable paradigm.

Difficulty-Controllable Question Generation DCQG is a relatively new task. Gao et al. (2019) classified questions as easy or hard according to whether they could be correctly answered by a BERT-based QA model, and controlled the question difficulty by modifying the hidden states before decoding. Another research on QG for knowledge graphs (Kumar et al., 2019) estimated the question difficulty based on popularity of the named entity. They manipulated the generation process by incorporating the difficulty level into the input embedding of the Transformer-based decoder. In our work, we control the question difficulty based on the number of its reasoning hops, which is more explainable.

Question Rewriting It is another emerging trend in the recent researches, demonstrating benefits to both QG and QA tasks. With rewriting, QG models produced more complex questions by incorporating more context information into simple questions

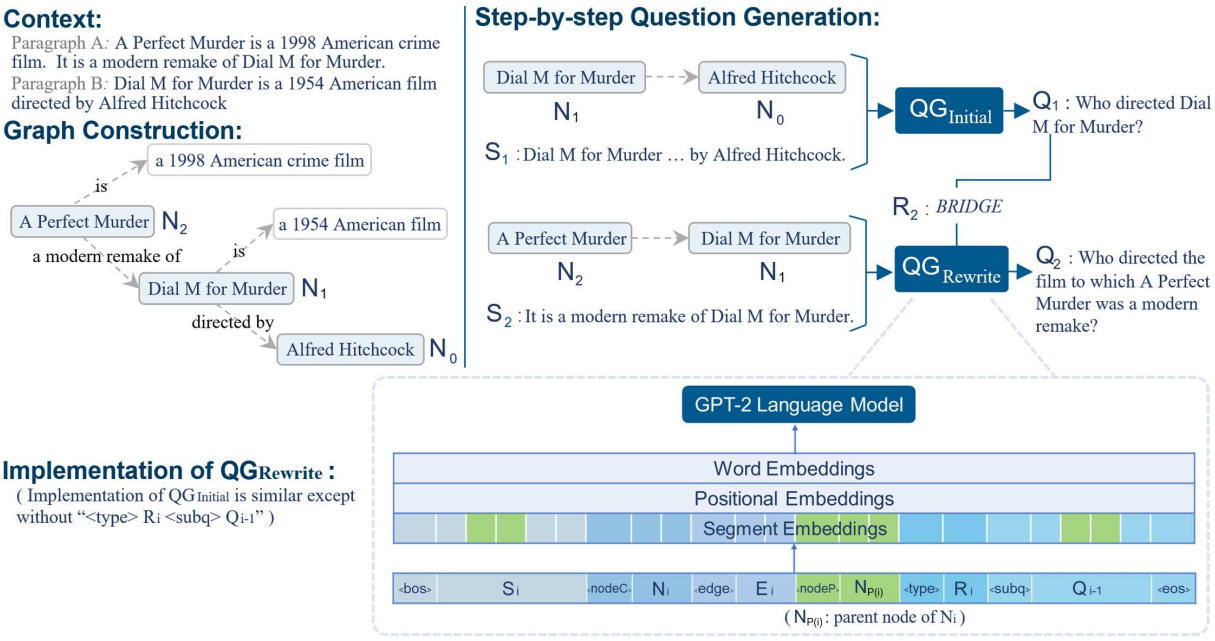


Figure 2: An overview of our proposed framework. The selected reasoning chain is marked as light blue nodes.

(Elgohary et al., 2019; Vakulenko et al., 2020), and QA pipelines could also decompose the original complex question into multiple shorter questions to improve model performance (Min et al., 2019; Khot et al., 2020).

3 Method

Given input context text $\mathcal{C}$ and a specific difficulty level d, our objective is to generate a *(question, answer)* pair $(\mathcal{Q}, \mathcal{A})$, where $\mathcal{A}$ is a sub-span of $\mathcal{C}$ and $\mathcal{Q}$ requires d-hop reasoning to answer. Fig. 2 and Algorithm 1 give an overview of our proposed framework. First, we construct a context graph $\mathcal{G}_{CG}$ corresponding to the given context, from which a subgraph $\mathcal{G}_T$ is selected to serve as the reasoning chain of the generated question. Next, with the reasoning chain and other contextual information as input, a question generator ($\text{QG}_{\text{Initial}}$) produces an initial simple question $\mathcal{Q}_1$. Then, $\mathcal{Q}_1$ is fed to a question rewriting module ($\text{QG}_{\text{Rewrite}}$), which iteratively rewrites it into a more complex question $\mathcal{Q}_i$ $(i = 2, 3, \ldots, d)$. In what follows, we will introduce the whole generation process in more details.

Context Graph Construction We follow the method proposed by Fan et al. (2019) to build the context graph $\mathcal{G}_{CG}$. Specifically, we first apply open information extraction (Stanovsky et al., 2018) to extract $\langle subject, relation, object \rangle$ triples from context sentences. Each triple is then transformed into two nodes connected with a directed edge, like *A Perfect Murder* $\xrightarrow{is}$ *a 1998 American crime film* in Fig. 2. The two nodes respectively represent the subject and object, and the edge describes their relation. Coreference resolution (Lee et al., 2017) is applied to merge nodes referring to the same entity. For instance, *A Perfect Murder* is merged with *It* in Fig. 2.

Reasoning Chain Selection With the context graph constructed, we sample a connected subgraph $\mathcal{G}_T$ consisting of $d+1$ nodes from it to serve as the reasoning chain of the generated question. A node $\mathcal{N}_0$ is first sampled as the answer of the question, if it is, or linked with, a named entity that has more than one node degree. Next, we extract from $\mathcal{G}_{CG}$ a maximum spanning tree $\mathcal{G}_L$, with $\mathcal{N}_0$ as its root node, e.g., the tree structure shown in Fig. 1. $\mathcal{G}_{CG}$ is temporarily considered as an undirected graph at this step. We then prune $\mathcal{G}_L$ into $\mathcal{G}_T$ to keep only $d+1$ nodes. During pruning, we consider the sentence position where each node is extracted in order to make the reasoning chain relevant to more context. In the following, we will denote a node in $\mathcal{G}_T$ as $\mathcal{N}_i$ $(i = 0, 1, \ldots, d)$, where each node is subscripted by *preorder traversal* of $\mathcal{G}_T$, and $\mathcal{N}_{P(i)}$ as the parent of $\mathcal{N}_i$.

Step-by-step Question Generation Our step-by-step QG process is described at lines 5-11 in Algorithm 1. The following notations are defined for clearer illustration:

Algorithm 1 Procedure of Our DCQG Framework

Input: context $\mathcal{C}$, difficulty level d
Output: $(\mathcal{Q}, \mathcal{A})$

$\mathcal{G}_{CG} \leftarrow \mathbf{BuildCG}(\mathcal{C})$
$\mathcal{N}_0 \leftarrow \mathbf{SampleAnswerNode}(\mathcal{G}_{CG})$
$\mathcal{G}_L \leftarrow \mathbf{MaxTree}(\mathcal{G}_{CG}, \mathcal{N}_0)$
$\mathcal{G}_T \leftarrow \mathbf{Prune}(\mathcal{G}_L, d)$
for $\mathcal{N}_i$ **in** $\mathbf{PreorderTraversal}(\mathcal{G}_T)$ **do**
  if $i = 0$ **then** continue
  $\mathcal{N}_{P(i)} = \mathbf{Parent}(\mathcal{N}_i)$
  $\mathcal{S}_i = \mathbf{ContextSentence}(\mathcal{C}, \mathcal{N}_i, \mathcal{N}_{P(i)})$
  $\mathcal{R}_i \leftarrow \begin{cases} \textit{Bridge} & \text{if } \mathcal{N}_i = \mathbf{FirstChild}(\mathcal{N}_{P(i)}) \\ \textit{Intersection} & \text{else} \end{cases}$
  $\mathcal{Q}_i \leftarrow \begin{cases} \mathbf{QG}_{Initial}(\mathcal{N}_i, \mathcal{N}_{P(i)}, \mathcal{S}_i) & \text{if } i = 1 \\ \mathbf{QG}_{Rewrite}(\mathcal{Q}_{i-1}, \mathcal{N}_i, \mathcal{N}_{P(i)}, \mathcal{S}_i, \mathcal{R}_i) & \text{else} \end{cases}$
end for
return $(\mathcal{Q}_d, \mathcal{N}_0)$

- $\mathcal{Q}_i$ $(i = 1, 2, \ldots, d)$ represents the question generated at each step, where $\mathcal{Q}_d$ is the final question $\mathcal{Q}$, and $\mathcal{Q}_{i+1}$ is rewritten from $\mathcal{Q}_i$ by adding one more hop of reasoning.
- $\mathcal{S}_i$ represents the context sentence from which we extract the triple $\mathcal{N}_i \rightarrow \mathcal{N}_{P(i)}$.
- $\mathcal{R}_i$ is the rewriting type of $\mathcal{Q}_i$ $(i = 2, 3, \ldots, d)$. Specifically, we consider two types of rewriting patterns in this work: *Bridge* and *Intersection*. As shown in Fig. 1, *Bridge*-style rewriting replaces an entity with a modified clause, while *Intersection* adds another restriction to an existing entity in the question. These two types can be distinguished by whether $\mathcal{N}_i$ is the first child of its parent node, i.e., whether its parent node has already been rewritten once in *Bridge* style.

To generate the final question with the required difficulty level d, we first use a question generator $\text{QG}_{\text{Initial}}$ to generate an initial simple question based on $\mathcal{N}_1$, $\mathcal{N}_0$, and the corresponding context sentence $\mathcal{S}_1$. Then, we repeatedly (for $d-1$ times) use $\text{QG}_{\text{Rewrite}}$ to rewrite question $\mathcal{Q}_{i-1}$ into a more complex one $\mathcal{Q}_i$, based on node $\mathcal{N}_i$ and its parent node $\mathcal{N}_{P(i)}$, context sentence $\mathcal{S}_i$, and the rewriting type $\mathcal{R}_i$ $(i = 2, 3, \ldots, d)$. Formally, the generation process of $\text{QG}_{\text{Initial}}$ and the rewriting process of $\text{QG}_{\text{Rewrite}}$ can be defined as:

$$\mathcal{Q}_1 = \arg\max_{\bar{\mathcal{Q}}_1} P(\bar{\mathcal{Q}}_1 | \mathcal{N}_1, \mathcal{N}_0, \mathcal{S}_1)$$

$$\mathcal{Q}_i = \arg\max_{\bar{\mathcal{Q}}_i} P(\bar{\mathcal{Q}}_i | \mathcal{Q}_{i-1}, \mathcal{N}_i, \mathcal{N}_{P(i)}, \mathcal{S}_i, \mathcal{R}_i)$$

where $i = 2, 3, \ldots, d$.

Algorithm 2 Procedure of Data Construction

Input: context $\mathcal{C} = \{\mathcal{P}_1, \mathcal{P}_2\}$, QA pair $(\mathcal{Q}_2, \mathcal{A}_2)$, supporting facts $\mathcal{F}$
Output: $\mathcal{R}_1, (\mathcal{Q}_1, \mathcal{A}_1), \mathcal{S}_1, \mathcal{S}_2, \{\mathcal{N}_0, \mathcal{E}_1, \mathcal{N}_1, \mathcal{E}_2, \mathcal{N}_2\}$

$\mathcal{R}_1 \leftarrow \mathbf{TypeClassify}(\mathcal{Q}_2)$
if $\mathcal{R}_1 \notin \{\textit{Bridge}, \textit{Intersection}\}$ **then** return
$subq_1, subq_2 \leftarrow \mathbf{DecompQ}(\mathcal{Q}_2)$
$suba_1, suba_2 \leftarrow \mathbf{QA}(subq_1), \mathbf{QA}(subq_2)$
$\mathcal{Q}_1, \mathcal{A}_1 \leftarrow \begin{cases} subq_2, suba_2 & \text{if } \mathcal{A}_2 = suba_2 \\ subq_1, suba_1 & \text{else} \end{cases}$
$\mathcal{S}_1, \mathcal{S}_2 \leftarrow \begin{cases} \mathcal{F} \cap \mathcal{P}_1, \mathcal{F} \cap \mathcal{P}_2 & \text{if } \mathcal{Q}_1 \text{ concerns } \mathcal{P}_1 \\ \mathcal{F} \cap \mathcal{P}_2, \mathcal{F} \cap \mathcal{P}_1 & \text{else} \end{cases}$
$\mathcal{N}_2 \leftarrow \mathbf{FindNode}(\mathcal{A}_2)$
$\mathcal{N}_0, \mathcal{E}_1, \mathcal{N}_1, \mathcal{E}_2 \leftarrow \mathbf{Match}(subq_1, subq_2)$

In our implementation, both $\text{QG}_{\text{Initial}}$ and $\text{QG}_{\text{Rewrite}}$ are initialized with the pre-trained GPT2-small model (Radford et al., 2019), and then fine-tuned on our constructed dataset (see Sec. 4). The encoder of $\text{QG}_{\text{Rewrite}}$, as illustrated in Fig. 2, is similar to Liu et al. (2020). If $\mathcal{N}_i$ points to $\mathcal{N}_{P(i)}$, then the input sequence is organized in the form of "$\langle bos \rangle$ $\mathcal{S}_i$ $\langle nodeC \rangle$ $\mathcal{N}_i$ $\langle edge \rangle$ $\mathcal{E}_i$ $\langle nodeP \rangle$ $\mathcal{N}_{P(i)}$ $\langle type \rangle$ $\mathcal{R}_i$ $\langle subq \rangle$ $\mathcal{Q}_{i-1}$ $\langle eos \rangle$", where $\mathcal{E}_i$ is the edge from $\mathcal{N}_i$ to $\mathcal{N}_{P(i)}$. The positions of "$\langle nodeC \rangle$ $\mathcal{N}_i$" and "$\langle nodeP \rangle$ $\mathcal{N}_{P(i)}$" will be exchanged if $\mathcal{N}_{P(i)}$ points to $\mathcal{N}_i$. As for $\text{QG}_{\text{Initial}}$, its input is organized in the same way except without "$\langle \text{type} \rangle$ $\mathcal{R}_i$ $\langle \text{subq} \rangle$ $\mathcal{Q}_{i-1}$".

The segment embedding layer is utilized to identify different segments. For those parts in $\mathcal{S}_i$ and $\mathcal{Q}_{i-1}$ that are the same as, or refer to the same entity as $\mathcal{N}_{P(i)}$, we replace their segment embeddings with the one of $\mathcal{N}_{P(i)}$, considering that the parent node of $\mathcal{N}_i$ plays an important role in denoting what to ask about, or which part to rewrite, as shown in Fig. 1.

4 Automatic Dataset Construction

Manually constructing a new dataset for our task is difficult and costly. Instead, we propose to automatically build a dataset from existing QA datasets without extra human annotation. In our work, the training data is constructed from HotpotQA (Yang et al., 2018), in which every context $\mathcal{C}$ consists of two paragraphs $\{\mathcal{P}_1, \mathcal{P}_2\}$, and most of the questions *require two hops of reasoning, each concerning one paragraph*. HotpotQA also annotates supporting facts $\mathcal{F}$, which are the part of the context most relevant to the question. In addition to the information already available in HotpotQA, we also need the following information to train

$\text{QG}_{\text{Initial}}$ and $\text{QG}_{\text{Rewrite}}$: i) $(\mathcal{Q}_1, \mathcal{A}_1)$, the simple initial question and its answer, which are used to train $\text{QG}_{\text{Initial}}$; ii) $\mathcal{R}_2$, the type of rewriting from $\mathcal{Q}_1$ to $\mathcal{Q}_2$; iii) $\{\mathcal{N}_0, \mathcal{N}_1, \mathcal{N}_2\}$, the reasoning chain of $\mathcal{Q}_2$; and iv) $\mathcal{S}_i$ $(i = 1, 2)$, the context sentences where we extract $\mathcal{N}_0$, $\mathcal{N}_1$ and $\mathcal{N}_2$.

Algorithm 2 describes our procedure to obtain the above information. The construction process is facilitated with the help of a reasoning type classifier (**TypeClassify**) and a question decomposer (**DecompQ**), referring to Min et al. (2019). For each question in HotpotQA (i.e. $\mathcal{Q}_2$), we first distinguish its reasoning type, and filter out those that are not *Bridge* and *Intersection*. The reasoning type here corresponds to the rewriting type $\mathcal{R}_i$. Then, **DecompQ** decomposes $\mathcal{Q}_2$ into two sub-questions, $subq_1$ and $subq_2$, based on span prediction and linguistic rules. For example, the $\mathcal{Q}_2$ in Fig. 2 will be decomposed into $subq_1$=*"To which film A Perfect Murder was a modern remake?"*, and $subq_2$=*"Who directed Dial M for Murder?"*. After that, an off-the-shelf single-hop QA model (Min et al., 2019) is utilized to acquire the answer of the two sub-questions, which should be *"Dial M for Murder"* and *"Alfred Hitchcock"* in the example.

As for $\mathcal{Q}_1$, it is one of the sub-questions. When $\mathcal{Q}_2$ is of the *Intersection* type, $\mathcal{Q}_1$ can be either $subq_1$ or $subq_2$. For the *Bridge* type, it is the sub-question that shares the same answer as $\mathcal{A}_2$. For the example above, $\mathcal{Q}_1$ is $subq_2$ because $suba_2 = \mathcal{A}_2$. The context sentence $\mathcal{S}_i$ is supposed to provide supporting facts contained in the paragraph $\mathcal{F}$ that concerns $\mathcal{Q}_i$ $(i = 1, 2)$. For the reasoning chain, it is selected from the local context graph by first locating $\mathcal{N}_2$ and then finding $\mathcal{N}_0, \mathcal{N}_1$ through text matching with the two sub-questions.

5 Experiments

In the following experiments, we mainly evaluate the generation results of our proposed method when required to produce 1-hop and 2-hop questions, denoted as $\text{Ours}_{\text{1-hop}}$ and $\text{Ours}_{\text{2-hop}}$. In Sec. 5.2, we compare our method with a set of strong baselines using both automatic and human evaluations on question quality. In Sec. 5.3, we provide controllability analysis by manually evaluating their difficulty levels and testing the performance of QA systems in answering questions generated by different methods. In Sec. 5.4, we test the effect of our generated QA pairs on the performance of a multi-hop QA model in a data augmentation setting. In Sec. 5.5, we further analyze the extensibility of our method, i.e., its potential in generating questions that require reasoning of more than two hops. Our code and constructed dataset have been made publicly available to facilitate future research.[1]

5.1 Experimental Setup

Datasets The constructed dataset described in Sec. 4 consists of 57,397/6,072/6,072 samples for training/validation/test. For context graph construction, we use the coreference resolution toolkit from AllenNLP 1.0.0 (Lee et al., 2017) and the open information extraction toolkit provided by the Plasticity developer API.[2] The question decomposer and the reasoning type classifier follow the implementations of Min et al. (2019).

Baselines The following baselines are trained to generate the 2-hop questions in the datasets:

- **NQG++** (Zhou et al., 2017) is a seq2seq model based on bi-directional Gate Recurrent Unit (GRU), with features enriched by answer position and lexical information.
- **ASs2s** (Kim et al., 2019) is a seq2seq model based on Long Short-term Memory (LSTM), which separately encodes answer and context.
- **SRL-Graph** and **DP-Graph** (Pan et al., 2020) are two state-of-the-art QG systems. They encode graph-level and document-level information with an attention-based Graph Neural Network (GNN) and a bi-directional GRU, respectively. SRL-Graph constructs the semantic graph by semantic role labelling, and DP-Graph by dependency parsing.
- **GPT2** is a vanilla GPT2-based QG model. Its input is the concatenation of context and sampled answer. The position where the answer appears in the context segment is denoted in the segment embedding layer.

Implementation Details The baseline models are trained to directly produce the 2-hop questions, while $\text{QG}_{\text{Initial}}$ and $\text{QG}_{\text{Rewrite}}$ are respectively trained to generate 1-hop questions and rewrite 1-hop ones into 2-hop. $\text{QG}_{\text{Initial}}$, $\text{QG}_{\text{Rewrite}}$, and GPT2 are initialized with the GPT2-small model from the HuggingFace Transformer library (Wolf et al., 2019), and fine-tuned for 8, 10, and 7 epochs, respectively, with batch size of 16. We apply top-p nucleus sampling with p = 0.9 during decoding.

[1] https://tinyurl.com/19esunzz
[2] https://www.plasticity.ai/

Model	BLEU3	BLEU4	METEOR	CIDEr
NQG++	15.41	11.50	16.96	-
ASs2s	15.21	11.29	16.78	-
SRL-Graph	19.66	15.03	19.73	-
DP-Graph	19.87	15.23	20.10	1.40
GPT2	20.98	**15.59**	**24.19**	1.46
$\text{Ours}_{\text{2-hop}}$	**21.07**	15.26	19.99	**1.48**

Table 1: Automatic evaluation results of the baseline models and the 2-hop questions generated by our method ($\text{Ours}_{\text{2-hop}}$).

AdamW (Loshchilov and Hutter, 2017) is used as optimizer, with the initial learning rate set to be 6.25×10^{-5} and adaptively decays during training. For DP-Graph, we use their released model and code to perform the experiment. For the other three baselines, we directly refer to the experiment results reported in Pan et al. (2020). The performances of these baselines are compared under the same setting as in Pan et al. (2020), where each context is abbreviated to only include the supporting facts and the part that overlaps with the question. More implementation details can be found in our code and the supplementary materials.

5.2 Evaluation of Question Quality

Automatic Evaluation The automatic evaluation metrics are BLEU3, BLEU4 (Papineni et al., 2002), METEOR (Lavie and Agarwal, 2007), and CIDEr (Vedantam et al., 2015), which measure the similarity between the generation results and the reference questions in terms of n-grams. As the four baselines are trained to generate 2-hop questions only, we only compare them with $\text{Ours}_{\text{2-hop}}$. As shown in Table 1, we can see that $\text{Ours}_{\text{2-hop}}$ and GPT2 perform consistently better than the others. Though the performances of $\text{Ours}_{\text{2-hop}}$ and GPT2 are close in terms of automatic metrics, we observe that the questions generated by $\text{Ours}_{\text{2-hop}}$ are usually more well-formed, concise and answerable, as illustrated in Table 2. These advantages cannot be reflected through automatic evaluation.

Human Evaluation We randomly sample 200 questions respectively from DP-Graph, GPT2, $\text{Ours}_{\text{1-hop}}$, $\text{Ours}_{\text{2-hop}}$, as well as the reference 1-hop and 2-hop questions in the constructed dataset ($\text{Gold}_{\text{1-hop}}$, $\text{Gold}_{\text{2-hop}}$). The questions are manually evaluated by eight human annotators, who are graduate students, majoring in English Literature, Computer Science, or Electronic Engineering. They voluntarily offer to help without being compensated in any form. Before annotation, they are informed of the detailed annotation instruction with clear scoring examples. The generated questions are evaluated in the following four dimensions:

$\text{Ours}_{\text{2-hop}}$	GPT2
When was the first theatre director of African descent born?	When was the first theatre director of African descent to establish a national touring company in the UK born?
What play by Carrie Hamilton was run at the Goodman Theatre in 2002?	What play by Carrie Hamilton and Carol Burnett ran at the Goodman Theatre and on Broadway in 2002?
What was the review score for the album that has been reissued twice?	What was the review of the album that includes previously unreleased tracks by Guetta from its first major international release?

Table 2: Examples of generation results from $\text{Ours}_{\text{2-hop}}$ and GPT2

- **Well-formed**: It checks whether a question is semantically correct. Annotators are asked to mark a question as *yes*, *acceptable*, or *no*. *Acceptable* is selected if the question is not grammatically correct, but its meaning is still inferrable.
- **Concise**: It checks whether the QG models are overfitted, generating questions with redundant modifiers. The question is marked as *yes* if no single word can be deleted, *acceptable* if it is a little lengthy but still in a natural way, and *no* if it is abnormally verbose.
- **Answerable**: It checks whether a question is answerable according to the given context. The anonnotion is either *yes* or *no*.
- **Answer Matching**: It checks whether the given answer is the correct answer to the question. The anonnotion is either *yes* or *no*.

The results are shown in Table 3. Overall, we can see that $\text{Ours}_{\text{2-hop}}$ performs consistently better than DP-Graph and GPT2 across all metrics and comparable to the hand-crafted reference questions. Our method performs especially well in terms of *concise*, even better than the reference questions. For reference, the average word number of the questions generated by DP-Graph, GPT2, $\text{Ours}_{\text{2-hop}}$, and $\text{Gold}_{\text{2-hop}}$ are 19.32, 19.26, 17.18, 17.44, respectively. It demonstrates that the enriched graph information and our multi-stage rewriting mechanism indeed enhance the question structure and content. In comparison, we find that the questions generated by the two baselines tend to unreasonably pile too many modifiers and subordinate clauses.

Difficulty Level	Model	Well-formed			Concise			Answerable		Answer Matching	
		Yes	Acceptable	No	Yes	Acceptable	No	Yes	No	Yes	No
2-hop	DP-Graph	28%	41%	31%	41%	53%	6%	49%	51%	39%	61%
	GPT2	57%	34%	9%	47%	50%	3%	69%	31%	66%	34%
	$\text{Ours}_{\text{2-hop}}$	**74%**	19%	7%	**67%**	30%	3%	**78%**	22%	**69%**	31%
	$\text{Gold}_{\text{2-hop}}$	72%	22%	6%	56%	40%	4%	92%	8%	87%	13%
1-hop	$\text{Ours}_{\text{1-hop}}$	**46%**	46%	8%	**65%**	25%	10%	**81%**	19%	**72%**	28%
	$\text{Gold}_{\text{1-hop}}$	56%	39%	5%	80%	16%	4%	84%	16%	79%	21%

Table 3: Human evaluation results of question quality.

Model	Inference Steps			
	1-hop	2-hop	3-hop	>3-hop
DP-Graph	26.1%	55.1%	8.7%	10.1%
GPT2	23.3%	57.1%	13.2%	6.4%
$\text{Ours}_{\text{2-hop}}$	4.3%	**67.7%**	25.8%	2.2%
$\text{Ours}_{\text{1-hop}}$	**70.7%**	28.2%	1.1%	0.0%

Table 4: Human evaluation results of the number of inference steps required by the generated questions.

Test Set	BERT		RoBERTa	
	EM	F1	EM	F1
DP-Graph	0.436	0.615	0.552	0.678
GPT2	0.419	0.581	0.669	0.772
$\text{Ours}_{\text{2-hop}}$	0.295	0.381	0.506	0.663
$\text{Ours}_{\text{1-hop}}$	0.618	0.737	0.882	0.937

Table 5: Performance of BERT- and RoBERTa-based QA models on different generated QA datasets.

As for the 1-hop questions, $\text{Ours}_{\text{1-hop}}$ performs well in terms of *answerable* and *answer matching*, but not so competitive in terms of *well-formed*, mainly due to the limitation of its training data. As the 1-hop reference questions ($\text{Gold}_{\text{1-hop}}$) are automatically decomposed from the hand-crafted 2-hop questions, a significant portion (44%) of them have some grammatical errors, but most of them are still understandable despite that.

5.3 Controllability Analysis

Human Evaluation of Controllability For controllability analysis, we manually evaluate the numbers of inference steps involved in generated questions. DP-Graph and GPT2 are also evaluated for comparison. The results are shown in Table 4. 70.65% of $\text{Ours}_{\text{1-hop}}$ require one step of inference and 67.74% of $\text{Ours}_{\text{2-hop}}$ require two steps, proving that our framework can successfully control the number of inference steps of most generated questions. In comparison, DP-Graph and GPT2 are not difficulty-aware and their generated questions are more scattered in difficulty levels.

Difficulty Assessment with QA Systems For further assessment of question difficulty, we test the performance of QA models in answering questions generated by different models. Specifically, we utilize two off-the-shelf QA models provided by the HuggingFace Transformer library (Wolf et al., 2019), which are respectively initialized with BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019b), and then fine-tuned on SQuAD (Rajpurkar et al., 2016). We select those generated questions that are ensured to be paired with correct answers by the human evaluation described in Sec. 5.2, and test the performance of two QA models in answering them. The evaluation metrics include Exact Match (EM) and F1.

The results are shown in Table 5. We can see that questions generated by $\text{Ours}_{\text{2-hop}}$ are more difficult than $\text{Ours}_{\text{1-hop}}$ not only to humans (requiring more hops of reasoning), but also to the state-of-the-art QA models. In comparison, with a more scattered mix of 1-hop and 2-hop questions, the performances on DP-Graph and GPT2 are between $\text{Ours}_{\text{1-hop}}$ and $\text{Ours}_{\text{2-hop}}$. This result demonstrates that our method can controllably generate questions of different difficulty levels for QA systems and that inference steps can effectively model the question difficulty.

5.4 Boosting Multi-hop QA Performance

We further evaluate whether the generated QA pairs can boost QA performance through data augmentation. Specifically, we heuristically sample the answers and reasoning chains from the context graphs in our constructed dataset to generate 150,305 two-hop questions. As a comparison, we utilize GPT2 to generate the same amount of data with the same sampled answers and contextual sentences. Some low-quality questions are filtered out if their word

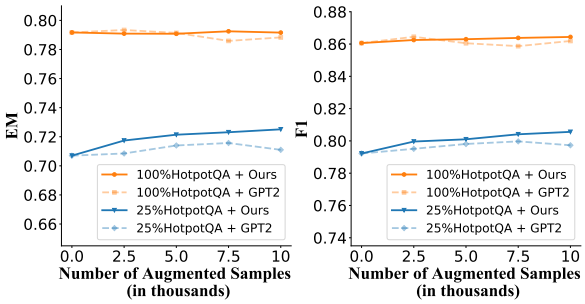


Figure 3: Performance of the DistilBERT-based QA system on HotpotQA, augmented with different quantities of generated data.

counts are not between 6∼30 (4.7% for ours and 9.2% for GPT2), or the answers directly appear in the questions (2.7% for ours and 2.4% for GPT2). Finally, we randomly sample 100,000 QA pairs and augment the HotpotQA dataset with them.

A DistilBERT-based (Sanh et al., 2019) QA model is implemented. It takes as input the concatenation of context and question to predict the answer span. To speed up the experiment, we only consider those necessary supporting facts as the question answering context. During training, the original samples from HotpotQA are oversampled to ensure that they are at least 4 times as the generated data. We use Adam (Kingma and Ba, 2015) as the optimizer, with the mini-batch size of 32. The learning rate is initially set to 3×10^{-5} and adaptively decays during training. The configurations are the same in all the QA experiments, except that the training datasets are different combinations of HotpotQA and the generated data. The validation and test sets are the same as those in HotpotQA.

We test the impact of the generated data under both *high-resource* (using the whole training set of HotpotQA) and *low-resource* settings (using only 25% of the data randomly sampled from HotpotQA). Fig. 3 compares the QA performance, augmented with different quantities of the data generated by our method and by GPT2, respectively. We can see that under both settings, our method achieves better performance than GPT2. Under the low-resource setting, performance boost achieved by our generated data is more significant and obviously better than that of GPT2. The performance of the QA model steadily improves when the training dataset is augmented with more data. EM and F1 of the QA model are improved by 2.56% and 1.69%, respectively, when 100,000 samples of our generated data are utilized.

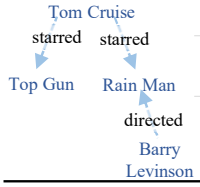
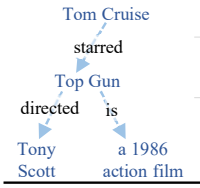
Reasoning Chain	QG Process	
	Q_1:	Which actor who starred in Top Gun?
	Q_2:	Which star of Top Gun was also in the movie Rain Man?
	Q_3:	Which star of Top Gun was also in the movie directed by Barry Levinson?
	Q_1:	Which actor who starred in Top Gun?
	Q_2:	What actor starred in the film that was directed by Tony Scott?
	Q_3:	What actor starred in the film that was directed by Tony Scott and was released in 1986?

Figure 4: Two examples of generating three-hop questions based on the extracted reasoning chains.

5.5 More-hop Question Generation

To analyze the extensibility of our method, we experiment with the generation of questions that are more than 2-hop, by repeatedly using $\text{QG}_{\text{Rewrite}}$ to increase question difficulty. Fig. 4 shows two examples of 3-hop question generation process. The two intermediate questions and the corresponding reasoning chains are also listed for reference.

We can see that the intermediate questions, serving as springboards, are effectively used by $\text{QG}_{\text{Rewrite}}$ to generate more complex questions. With the training data that only contains 1-hop and 2-hop questions, our framework is able to generate some high-quality 3-hop questions, demonstrating the extensibility of our framework. It can be expected that the performance of our model can be further strengthened if a small training set of 3-hop question data is available.

Besides, it can also be observed that though the contexts and answers of these two questions are the same, two different questions with different underlying logic are generated, illustrating that the extracted reasoning chain effectively controls the question content.

However, when generating questions with more than 3 hops, we find that the question quality drastically declines. The semantic errors become more popular, and some content tend to be unreasonably repeated. It is probably because the input of $\text{QG}_{\text{Rewrite}}$ has become too long to be precisely encoded by the GPT2-small model due to the growing length of the question. It will be our future work to explore how to effectively extend our method to more-hop question generation.

6 Conclusion

We explored the task of difficulty-controllable question generation, with question difficulty redefined as the inference steps required to answer it. A step-by-step generation framework was proposed to accomplish this objective, with an input sampler to extract the reasoning chain, a question generator to produce a simple question, and a question rewriter to further adapt it into a more complex one. A dataset was automatically constructed based on HotpotQA to facilitate the research. Extensive evaluations demonstrated that our method can effectively control difficulty of the generated questions, and keep high question quality at the same time.

Acknowledgments

Thanks to Zijing Ou, Yafei Liu and Suyuchen Wang for their helpful comments on this paper.

References

Zi Chai and Xiaojun Wan. 2020. Learning to ask more: Semi-autoregressive sequential question generation under dual-graph interaction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 225–237. 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. Association for Computational Linguistics.

Ahmed Elgohary, Denis Peskov, and Jordan L. Boyd-Graber. 2019. Can you unpack that? Learning to rewrite questions-in-context. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5917–5923. Association for Computational Linguistics.

Angela Fan, Claire Gardent, Chloé Braud, and Antoine Bordes. 2019. Using local knowledge graph construction to scale seq2seq models to multi-document inputs. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4184–4194. Association for Computational Linguistics.

Yifan Gao, Lidong Bing, Wang Chen, Michael R. Lyu, and Irwin King. 2019. Difficulty controllable generation of reading comprehension questions. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 4968–4974.

Junmo Kang, Haritz Puerto San Roman, and Sung-Hyon Myaeng. 2019. Let me know what to ask: Interrogative-word-aware question generation. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 163–171. Association for Computational Linguistics.

Tushar Khot, Daniel Khashabi, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2020. Text modular networks: Learning to decompose tasks in the language of existing models. *CoRR*, abs/2009.00751.

Yanghoon Kim, Hwanhee Lee, Joongbo Shin, and Kyomin Jung. 2019. Improving neural question generation using answer separation. In *The Thirty-Third AAAI Conference on Artificial Intelligence*, pages 6602–6609. AAAI Press.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*.

Vishwajeet Kumar, Yuncheng Hua, Ganesh Ramakrishnan, Guilin Qi, Lianli Gao, and Yuan-Fang Li. 2019. Difficulty-controllable multi-hop question generation from knowledge graphs. In *The 18th International Semantic Web Conference*, volume 11778 of *Lecture Notes in Computer Science*, pages 382–398. Springer.

Ghader Kurdi, Jared Leo, Bijan Parsia, Uli Sattler, and Salam Al-Emari. 2020. A systematic review of automatic question generation for educational purposes. *Int. J. Artif. Intell. Educ.*, 30(1):121–204.

Alon Lavie and Abhaya Agarwal. 2007. METEOR: an automatic metric for MT evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231. Association for Computational Linguistics.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197. Association for Computational Linguistics.

Bang Liu, Haojie Wei, Di Niu, Haolan Chen, and Yancheng He. 2020. Asking questions the human way: Scalable question-answer generation from text corpus. In *The Web Conference*, pages 2032–2043. ACM / IW3C2.

Bang Liu, Mingjun Zhao, Di Niu, Kunfeng Lai, Yancheng He, Haojie Wei, and Yu Xu. 2019a. Learning to generate questions by learningwhat not to generate. In *The World Wide Web Conference*, pages 1106–1118.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in Adam. *CoRR*, abs/1711.05101.

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. Multi-hop reading comprehension through question decomposition and rescoring. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, Volume 1: Long Papers*, pages 6097–6109. Association for Computational Linguistics.

Liangming Pan, Wenqiang Lei, Tat-Seng Chua, and Min-Yen Kan. 2019. Recent advances in neural question generation. *CoRR*, abs/1905.08949.

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. 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. ACL.

Judea Pearl and Azaria Paz. 1986. Graphoids: Graph-based logic for reasoning about relevance relations or when would x tell you more about y if you already know z? In *Advances in Artificial Intelligence II, Seventh European Conference on Artificial Intelligence, ECAI*, pages 357–363. North-Holland.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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. Association for Computational Linguistics.

Mrinmaya Sachan and Eric Xing. 2016. Easy questions first? A case study on curriculum learning for question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 453–463, Berlin, Germany. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Gabriel Stanovsky, Julian Michael, Luke Zettlemoyer, and Ido Dagan. 2018. Supervised open information extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 885–895. Association for Computational Linguistics.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 641–651. Association for Computational Linguistics.

Svitlana Vakulenko, Shayne Longpre, Zhucheng Tu, and Raviteja Anantha. 2020. Question rewriting for conversational question answering. *arXiv preprint arXiv:2004.14652*.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015. CIDEr: Consensus-based image description evaluation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4566–4575. IEEE Computer Society.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. 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. Association for Computational Linguistics.

Jianxing Yu, Xiaojun Quan, Qinliang Su, and Jian Yin. 2020. Generating multi-hop reasoning questions to improve machine reading comprehension. In *The Web Conference*, pages 281–291. ACM / IW3C2.

Yuyu Zhang, Xinshi Chen, Yuan Yang, Arun Ramamurthy, Bo Li, Yuan Qi, and Le Song. 2020. Efficient probabilistic logic reasoning with graph neural networks. In *8th International Conference on Learning Representations*. OpenReview.net.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. In *In Proceedings of 6th CCF International Conference on Natural Language*, volume 10619 of *Lecture Notes in Computer Science*, pages 662–671. Springer.

Kangli Zi, Xingwu Sun, Yanan Cao, Shi Wang, Xiaoming Feng, Zhaobo Ma, and Cungen Cao. 2019. Answer-focused and position-aware neural network

for transfer learning in question generation. In *Proceedings of the 12th International Conference of Knowledge Science, Engineering and Management*, volume 11776 of *Lecture Notes in Computer Science*, pages 339–352. Springer.