Locate Then Ask: Interpretable Stepwise Reasoning for Multi-hop Question Answering

Siyuan Wang[1], **Zhongyu Wei**[1,2]*, **Zhihao Fan**[1], **Qi Zhang**[3], **Xuanjing Huang**[3]
[1]School of Data Science, Fudan University, China
[2]Research Institute of Intelligent and Complex Systems, Fudan University, China
[3]School of Computer Science, Fudan University, China
{wangsy18,zywei,fanzh18,qz,xjhuang}@fudan.edu.cn

Abstract

Multi-hop reasoning requires aggregating multiple documents to answer a complex question. Existing methods usually decompose the multi-hop question into simpler single-hop questions to solve the problem for illustrating the explainable reasoning process. However, they ignore grounding on the supporting facts of each reasoning step, which tends to generate inaccurate decompositions. In this paper, we propose an interpretable stepwise reasoning framework to incorporate both single-hop supporting sentence identification and single-hop question generation at each intermediate step, and utilize the inference of the current hop for the next until reasoning out the final result. We employ a unified reader model for both intermediate hop reasoning and final hop inference and adopt joint optimization for more accurate and robust multi-hop reasoning. We conduct experiments on two benchmark datasets HotpotQA and 2WikiMultiHopQA. The results show that our method can effectively boost performance and also yields a better interpretable reasoning process without decomposition supervision. [1]

1 Introduction

Recent years have witnessed an emerging trend in the task of multi-hop question answering. It requires the model to aggregate multiple pieces of documents (i.e., context) and perform multi-hop reasoning to infer the answer (Talmor and Berant, 2018; Khashabi et al., 2018). Several datasets have been introduced as benchmarks, such as HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020) and WikiHop (Welbl et al., 2018), and the first two provide supporting facts supervision to encourage models to further explain what supporting sentences lead to the prediction.

The first generation of models for multi-hop question answering utilizes a one-step reader (Qiu et al., 2019; Fang et al., 2019; Shao et al., 2020; Tu et al., 2020; Beltagy et al., 2020) to capture the interaction between the question and relevant contexts for the prediction of the answer as well as the supporting sentences. In order to model the explainable multi-step reasoning process, researchers explore to decompose the multi-hop question into easier single-hop questions and solve sub-questions to reach the answer (Talmor and Berant, 2018; Wolfson et al., 2020).

Question decomposition based approaches achieve promising prediction performance and are able to demonstrate the reasoning process to some

[Question]
Q: What city is the Marine Air Control Group 28 located in?
[Context]
P1: Marine Tactical Air Command Squadron 28 is a United States Marine Corps aviation command and control unit based at Marine Corps Air Station Cherry Point. They provide the 2nd Marine Aircraft Wings tactical headquarters and …
P2: Marine Corps Air Station Cherry Point or MCAS Cherry Point is a United States Marine Corps airfield located in **Havelock**, North Carolina, United States, in the eastern part of the state. It ...
P3-P10: …

[Question Decomposition (Min et al., 2019)**]**
Sub-Q1: Which Marine Air Control Group 28?
Sub-Q2: What city is Marine Tactical Air Command Squadron 28 located in?

[Stepwise Decomposition]
Step1-S: Marine Tactical Air Command Squadron 28 is a … control unit based at Marine Corps Air Station Cherry Point.
Step1-Q: Which is the base of Marine Air Control Group 28?

Step2-S: Marine Corps Air Station Cherry Point ... United States Marine Corps airfield located in Havelock, North Carolina … state.
Step2-Q: What city is the Marine Corps Air Station Cherry Point located in?

Figure 1: A multi-hop reasoning example from HotpotQA. To solve the problem, DecompRC (Min et al., 2019) generates improper decomposition of questions and predicts a wrong answer while our expected stepwise decomposition includes both single-hop supporting sentences and sub-questions of each step to reason out the correct answer. The underlined phrase is the fact uncovered by machine-generated decomposition while the shaded contexts support the corresponding single-hop question generation.

*Corresponding author
[1]Codes are publicly available at https://github.com/WangsyGit/StepwiseQA.

Proceedings of the 29th International Conference on Computational Linguistic, pages 1655–1665
October 12–17, 2022.

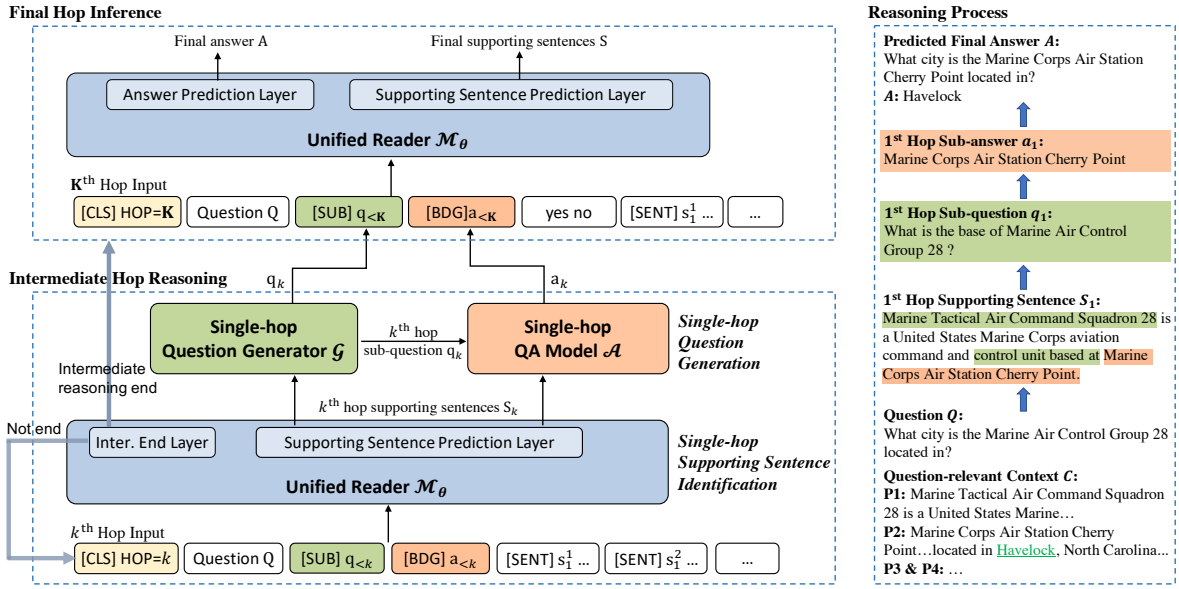

Figure 2: The overall architecture of stepwise reasoning framework with an interpretable reasoning process.

extent. However, the single-hop questions are generated solely based on the original question without considering the supporting facts each step involves (Min et al., 2019; Perez et al., 2020; Khot et al., 2020). This usually leads to wrong-guided decomposition and inaccurate explanations. An example is shown in Figure 1 including a question, two relevant contextual paragraphs, two sub-questions generated by Min et al. (2019) and two expected steps of reasoning with supporting sentences and sub-questions. The first single-hop question generated by an existing model (*Sub-Q1*) fails to query "the base of Marine Air Control Group 28" which is beyond the scope of the original multi-hop question and such an improperly reasoned hop also leads to the failure of final answer prediction. We argue that a complete step of reasoning should consist of intermediate supporting sentence identification and sub-question generation to reduce the inference error in the procedure.

In this paper, we propose a stepwise reasoning framework for multi-hop question answering. It performs both single-hop supporting sentence identification and single-hop question generation in each step, and reasons from one intermediate hop to the next until the final hop inference. Specifically, we perform an intermediate hop reasoning that locates the single-hop supporting sentences and constructs the sub-question based on the original question and the corresponding supporting facts in each step. We utilize an off-the-shelf single-hop question generator to eliminate the need for human annotations and avoid the risk of noisy labels posed by constructed pseudo-supervision. In the final hop, we simultaneously predict the answer and the supporting sentences of the multi-hop question according to the preceding hops. We employ a unified reader model for both intermediate single-hop supporting sentence identification and final hop inference and jointly learn them so that a midway error may be corrected by subsequent hops to mitigate cumulative failures. We further adopt two measures to reduce the train-test discrepancy of single-hop supporting sentences and sub-questions to mitigate exposure bias for better generalization.

Experiments are conducted on two benchmark datasets involving different hops of reasoning, HotpotQA (Yang et al., 2018) and 2WikiMultiHopQA (Ho et al., 2020). The results indicate that our stepwise reasoning framework achieves significant improvements and shows general effectiveness across different reasoning types. Further analysis and qualitative cases also demonstrate that our method generates high-quality single-hop questions for interpretable multi-hop reasoning.

2 Methodology

Given a multi-hop question Q and a context including multiple paragraphs, we aim to read the question-relevant context C to predict the final answer A and explain it with the supporting sentences S. As illustrated in Figure 2, we present a stepwise reasoning framework to iteratively identify the single-hop supporting sentences and generate

the single-hop question for the following reasoning, which consists of three components as below. It first filters out the unrelated paragraphs to extract the question-relevant context C (§ 2.1). Then it identifies the supporting sentences of each intermediate hop from the relevant context to ask and answer the corresponding single-hop question, and passes the auto-generated messages to the next hop (§ 2.2). After intermediate hop reasoning ends, the last module predicts the final answer and the supporting sentences of the multi-hop question according to the preceding inference (§ 2.3). We jointly train a unified reader model for all reasoning hops (§ 2.4) and adopt two measures to mitigate the train-test discrepancy for better inference (§ 2.5).

2.1 Context Filter

In order to reduce the distraction in the context for downstream multi-hop reasoning process, we select the most relevant paragraphs as the question-relevant context C. We first train a paragraph selection model which takes the question and the concatenation of all candidate paragraphs as the input and predicts the probability scores that each paragraph is relevant to answer the question.

As HotpotQA mainly consists of 2-hop questions which involve two relevant paragraphs, we follow the 2-hop selection strategy in (Fang et al., 2019). For the first hop, it selects the paragraph with the highest scores among the paragraphs containing the same phrases as the question. Then the second-hop paragraph is extracted by hyperlinks from the first selected one. Two other paragraphs with the next highest scores are also selected to constitute the question-relevant context. For 2WikiMultiHopQA dataset with 2∼4 hop questions, we select five paragraphs with the highest scores and filter the other paragraphs from the context.

2.2 Intermediate Hop Reasoning

Based upon the filtered relevant context, we perform the multi-hop reasoning step-by-step. We adopt a unified reader model $\mathcal{M}_\theta$ to iteratively identify the single-hop supporting sentences focused at each intermediate hop, and decide whether to end the intermediate hop reasoning indicating that the cumulative intermediate information is ready for final hop inference. Then depending on the predicted supporting sentences at each hop, the corresponding single-hop question can be generated and answered, which will be passed to the reader for next hop reasoning. This iterative process is repeated until the intermediate reasoning is ended, or up to $K-1$ intermediate hops.

Single-hop Supporting Sentence Identification The reader attempts to find the supporting sentences at each hop $k \in \{1, ..., K-1\}$ given the concatenation of the original question Q, the sub question-answer pairs $\{(q_1, a_1)\ ...\ (q_{k-1}, a_{k-1})\}$ of previous hops, and the relevant context C as the input. Specifically, the concatenated sequence is formulated as `[CLS] HOP=k [SEP] Q` `[SUB]` q_1 `[BDG]` a_1 `...` `[SUB]` q_{k-1} `[BDG]` a_{k-1} `[SEP] [SENT]` s_1^1 `[SENT]` s_2^1 `...[SEP] [SENT]` s_1^2 `...[SEP]` where `HOP=k` indicates the current hop number. Special tokens `[SUB]`, `[BDG]`, `[SENT]` respectively represent the single-hop sub-question, sub-answer and each sentence, and s_j^i is the j-th sentence of the i-th paragraph in the relevant context C.

On top of the representations of each sentence token `[SENT]`, we build a binary classifier to predict the probability $p_{i,j}^{(k)}$ that each sentence s_j^i is a supporting fact of the current hop. The corresponding binary cross entropy loss $\mathcal{L}_{sf}^{(k)}$ is calculated as Eq. 1. $y_{i,j}^{(k)}$ is the label whether the sentence s_j^i is a supporting fact of the hop k, and N_s is the total number of sentences in C.

$$\mathcal{L}_{sf}^{(k)} = \frac{1}{N_s}\sum_i\sum_j -y_{i,j}^{(k)}\log(p_{i,j}^{(k)}) - (1-y_{i,j}^{(k)})\log(1-p_{i,j}^{(k)}) \quad (1)$$

Then the `[CLS]` representation is also fed into a binary classifier to compute the probability $p_{end}^{(k)}$ that the intermediate reasoning should be ended at hop k and go on to final hop inference, and the cross entropy loss is as Eq. 2. $y_{end}^{(k)}$ is the label whether to end the intermediate hop reasoning at current hop k.

$$\mathcal{L}_{end}^{(k)} = -y_{end}^{(k)}\log(p_{end}^{(k)}) - (1-y_{end}^{(k)})\log(1-p_{end}^{(k)}) \quad (2)$$

Single-hop Question Generation After identifying the supporting sentences S_k of hop k, we generate the corresponding single-hop question q_k to investigate what the current hop asks about. In this work, we do not use annotated or pseudo supervision to train a question decomposition model. Instead, we take inspiration from (Pan et al., 2020) and adopt a pre-trained simple question generator

$\mathcal{G}_s$ to directly output the desired single-hop question, which is trained beforehand on top of an off-the-shelf simple question corpus.

To encourage the single-hop question more grounded on the contextual facts, we generate them based on both the identified single-hop supporting sentences S_k and the multi-hop question Q. The latter serves as a guidance for the generation towards the original reasoning goal. Specifically, we extract the intersectional tokens between S_k and Q as the prompt and append it to the supporting sentences S_k as the input for single-hop question generation, which is organized as `[CLS]` (Q $\cap$ $\mathrm{S_k}$) `[SEP]` $\mathrm{S_k}$ `[SEP]`. Correspondingly, during the single-hop question generator pre-training, we also take a single sentence as the context and utilize the tokens existing within both the target question and the context to prompt simple question generation.

Queried with the generated single-hop question q_k, we immediately resolve it to ease the whole multi-hop question. We also leverage the aforementioned simple question dataset to pre-train a single-hop QA model $\mathcal{A}$ to make it more consistent with the single-hop question generator. Then according to the single-hop supporting sentences S_k and question q_k at hop k, we utilize the pre-trained QA model $\mathcal{A}$ to output the single-hop answer a_k, which together with the single-hop question q_k will be passed to the next hop for subsequent reasoning.

2.3 Final Hop Inference

After the end of intermediate hop reasoning, we can utilize the single-hop questions and answers $\{(q_1, a_1)$ ... $(q_{K-1}, a_{K-1})\}$ of all previous hops to build a bridge for inferring the final hop K. We employ the same unified reader model $\mathcal{M}_\theta$ during intermediate hop reasoning to predict the final answer A of the multi-hop question Q and simultaneously provide the overall explanatory supporting sentences S. The input sequence fed into the reader for the final hop is similar to intermediate hops, except that we insert two additional tokens `yes` and `no` before the relevant context C for answer prediction. As there are three answer types (yes/no/span), we integrate the answer type classification into the answer span prediction by appending `yes` and `no` as two candidate spans. We reformulate the current input as `[CLS] HOP=K [SEP] Q [SUB]` q_1 `[BDG]` a_1 `...` `[SUB]` q_{K-1} `[BDG]` a_{K-1} `[SEP]` **`yes no`** `[SEP] [SENT]` s_1^1 `...[SEP] [SENT]` s_1^2 `...[SEP]`.

To accomplish the final hop inference, we first utilize the same binary classifier to identify whether each sentence is a supporting fact of the whole multi-hop question, and compute a final supporting sentence identification loss $\mathcal{L}_{sf}^{(K)}$. Then for the final answer span prediction, we attach a linear layer with a softmax function to all context representations to obtain the probability of each token t_n being the start p_n^s or end position p_n^e of the answer span. The cross entropy loss is calculated as following formula, where token t_x and t_y are respectively the labels of start or end positions.

$$\mathcal{L}_{span} = -\log(p_x^s) - \log(p_y^e) \tag{3}$$

2.4 Optimization & Inference

In order to optimize our framework, we can first set up a maximum number of required reasoning hops K. Then our stepwise reasoning framework essentially comprises $K-1$ iterative intermediate hop reasoning layers and a final hop inference layer. As there is no previous single-hop question-answer pair before the first hop reasoning, we omit them within the initial input which will be fed into the first intermediate hop reasoning layer. We jointly train our stepwise reasoning framework for all intermediate hops and the final hop in order that an intermediate mistake can be corrected by subsequent hops to mitigate cascading failures. All losses are combined in a weighted manner:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{sf}^{int} + \lambda_2 \mathcal{L}_{end}^{int} + \lambda_3 \mathcal{L}_{sf}^{(K)} + \mathcal{L}_{span} \tag{4}$$

$$\mathcal{L}_{sf}^{int} = \frac{\mathcal{L}_{sf}^{(1)} + \sum_{k=2}^{K-1}(1 - y_{end}^{(k-1)})\mathcal{L}_{sf}^{(k)}}{1 + \sum_{k=2}^{K-1}(1 - y_{end}^{(k-1)})} \tag{5}$$

$$\mathcal{L}_{end}^{int} = \frac{1}{k_e}\sum_{k=1}^{k_e}\mathcal{L}_{end}^{(k)}, \text{where } y_{end}^{(k_e)} = 1 \tag{6}$$

where λ_1, λ_2 and λ_3 are weighted hyper-parameters. $\mathcal{L}_{sf}^{int}$ is the average supporting sentence identification loss of all actual intermediate hops that are not ended. $\mathcal{L}_{end}^{int}$ is the average loss of intermediate reasoning end prediction.

During the inference period, we start from the first hop and dynamically reason from one hop to the next until the final hop. We predict whether to end the intermediate reasoning at each intermediate hop. Once it is over, we utilize all generated sub-questions and sub-answers and move to conduct the final hop inference. If not end, we will pass them

to the next intermediate hop and repeat the process until the intermediate hop reasoning is ended, or until reaching $K-1$ intermediate hops.

2.5 Exposure Bias Mitigation

In light of the design of our stepwise reasoning framework, there may arise the exposure bias problem between optimization and inference. Given the ground-truth supporting sentence supervision for each intermediate hop S_k^t, we can generate the target single-hop question and answer for the follow-up reasoning at training time. However, at test time we can only conduct single-hop question generation based on the predicted single-hop supporting sentences S_k which may deviate from the oracle ones S_k^t. To address this, we propose two measures to respectively reduce the discrepancy between train-test single-hop supporting sentences and train-test single-hop questions.

Firstly, we train a separate reader model only for the intermediate single-hop supporting sentence identification, and adopt it to re-predict the single-hop supporting sentences of training data with occasionally injected mistakes for optimizing the whole framework. Thereby we can regulate bias between training and test supporting sentences of intermediate hops. Besides, we also augment the training data for the single-hop question generation $\mathcal{G}_s$ by taking the re-predicted training single-hop supporting sentences S_k as input, and the generated sub-questions based on the ground-truth supporting sentences S_k^t as the target. Then the generator is trained to recover from the non-gold single-hop supporting sentences to approximate the oracle ones and reduce the deviation between train-test intermediate single-hop questions. With these two strategies, we can jointly optimize our stepwise reasoning framework for better generalization to non-golden test cases.

3 Experiments

3.1 Experimental Dataset

We take two datasets HotpotQA (Yang et al., 2018) and 2WikiMultiHopQA (Ho et al., 2020) that involve different reasoning hops as a testbed to study textual multi-hop reasoning. They both require answering the question as well as predicting the supporting facts to explain the reasoning. HotpotQA includes both distractor setting and fullwiki setting, and we focus on the former with limited candidate paragraphs to fully test the multi-hop reasoning ability while putting aside the information retrieval part. Although 2WikiMultiHopQA provides annotated evidence for interpreting the reasoning path, we leave them out of account to illustrate the effectiveness and interpretability of our framework without explanation supervision.

HotpotQA and 2WikiMultiHopQA respectively consist of $90,447/7,405/7,405$ and $167,454/12,576/12,576$ samples in training, development and test sets, and each instance is provided with 10 paragraphs. HotpotQA comprises 2-hop questions that only two paragraphs contain necessary supporting sentences, while 2WikiMultiHopQA contains 2$\sim$4 hop questions and the supporting facts reside in two to four paragraphs. Besides, to train the single-hop question generator and QA model, we use SQuAD (Rajpurkar et al., 2016) as the simple question corpus.

3.2 Implementation Details

We take ELECTRA-large (Clark et al., 2020) as the backbone of our proposed framework and the single-hop QA model, and train a single-hop question generator using BART-large (Lewis et al., 2019). The weights to balance losses are chosen as $\lambda_1 = 10/5$ (for HotpotQA/2WikiMultiHopQA), $\lambda_2 = 2$ and $\lambda_3 = 5$. The maximum value of required reasoning hops is set as $K = 2$ for HotpotQA and $K = 4$ for 2WikiMultiHopQA. More training details are given in Appendix A.

3.3 Overall Performance

We compare our **step**wise **reasoner** (*StepReasoner*) with previous published methods on HotpotQA and 2WikiMultiHopQA. Since there are few systems on the leaderboard of 2WikiMultiHopQA, we follow (Fu et al., 2021) and make a comparison with more models on both dev and test sets. The compared methods cover both question decomposition based models and one-step reading based models.

Question decomposition based models include *DecompRC* (Min et al., 2019), *ONUS* (Perez et al., 2020), *QFE* (Nishida et al., 2019), *CRERC* (Fu et al., 2021)) and *NA-Reviewer* (Fu et al., 2022).

One-step reading based models consist of *DFGN* (Qiu et al., 2019), *ELECTRA* (Clark et al., 2020), *TAP2* (Glass et al., 2019), *SAE-large* (Tu et al., 2020), *C2F Reader* (Shao et al., 2020), *Longformer* (Beltagy et al., 2020), *ETC-large* (Zaheer et al., 2020), *FFReader-large* (Alkhaldi et al., 2021), *HGN-large* (Fang et al., 2019).

Model	Answer		Sup Fact		Joint	
	EM	F1	EM	F1	EM	F1
DecompRC (Min et al., 2019)	55.20	69.63	-	-	-	-
ONUS (Perez et al., 2020)	66.33	79.34	-	-	-	-
TAP2 (Glass et al., 2019)	64.99	78.59	55.47	85.57	39.77	69.12
SAE-large (Tu et al., 2020)	66.92	79.62	61.53	86.86	45.36	71.45
C2F Reader (Shao et al., 2020)	67.98	81.24	60.81	87.63	44.67	72.73
Longformer (Beltagy et al., 2020)	68.00	81.25	63.09	88.34	45.91	73.16
ETC-large (Zaheer et al., 2020)	68.12	81.18	**63.25**	**89.09**	46.40	73.62
FFReader-large (Alkhaldi et al., 2021)	68.89	82.16	62.10	88.42	45.61	73.78
HGN-large (Fang et al., 2019)	69.22	82.19	62.76	88.47	47.11	74.21
StepReasoner	**69.66**	**82.42**	62.99	87.85	**47.84**	**74.27**

Table 1: Experimental results of different models on the test set of HotpotQA distractor setting.

Model	Answer		Sup Fact	
	EM	F1	EM	F1
Dev				
DecompRC (Min et al., 2019)	7.46	41.57	56.49	82.73
QFE (Nishida et al., 2019)	37.56	43.21	21.13	59.20
CRERC (Fu et al., 2021)	71.56	74.51	**86.00**	**92.75**
NA-Reviewer (Fu et al., 2022)	76.88	82.30	-	-
DFGN (Qiu et al., 2019)	30.87	38.49	17.06	57.79
ELECTRA-base	66.81	72.28	81.19	90.96
ELECTRA-large	79.22	83.51	83.08	92.01
StepReasoner(ELECTRA-base)	68.11	73.03	81.72	91.21
StepReasoner(ELECTRA-large)	**80.23**	**84.26**	83.41	92.01
Test				
HGN-revise (Fang et al., 2019)	71.20	75.69	69.35	89.07
CRERC (Fu et al., 2021)	69.58	72.33	82.86	90.68
NA-Reviewer (Fu et al., 2022)	76.73	81.91	**89.61**	**94.31**
StepReasoner	**80.88**	**84.86**	83.30	91.89

Table 2: Results on the dev and test sets of 2WikiMultiHopQA.

The results are shown in Table 1 and 2. We find that *StepReasoner* outperforms all models in terms of both answer prediction and joint evaluation and achieves comparable performance in supporting fact prediction, which demonstrates the effectiveness of our method. Specifically, it performs better than both question decomposition based and one-step reading based methods. The former improvement indicates that a unified reader to stepwise identify the single-hop supporting sentences for single-hop sub-question generation can enhance the accuracy of previous question decomposition methods. The latter verifies that the interpretability injected by stepwise reasoning can also improve the QA performance. Besides, *Longformer*, *ETC-large*, *FFReader-large* and *HGN-large* show a better supporting fact prediction performance on HotpotQA, especially in F1 score. This is because the first three models are designed for handling longer sequences and the last utilizes a complex hierarchical graph network which both can cover more candidate paragraphs for supporting sentence prediction. For 2WikiMultiHopQA, *CRERC* and *NA-Reviewer* achieve better supporting fact prediction because they both utilize external annotated evidence for training which confirms the effectiveness of our *StepReasoner* without explanation supervision. We also evaluate the intermediate reasoning end prediction, and find that our *StepReasoner* can exactly decide when to end the intermediate hop reasoning.

3.4 Further Analysis

Ablation Study To dive into the sources of performance gain in our *StepReasoner*, we conduct an ablation study on the HotpotQA development set, which is shown in Table 3. Compared to the overall stepwise reasoning system *StepReasoner*, a pipeline model without joint training shows a sharp performance degradation. It indicates that joint optimization of a unified reader model for all hops can improve the tolerance for intermediate faults and boost reasoning performance. After removing any measures to mitigate exposure bias, the performance has also significantly dropped. It shows that both two measures to alleviate the train-test discrepancy of single-hop supporting sentences and single-hop questions confirm a better generalization to cases deviated from oracle.

Effectiveness on Various Backbone Models To analyze the effectiveness of the *StepReasoner* based on different backbones, we vary several pre-trained models of different scales including *BERT-base-uncased* (Liu et al., 2019), *ELECTRA-large*

Model	Answer		Sup Fact		Joint	
	EM	F1	EM	F1	EM	F1
StepReasoner	70.11	83.03	64.27	88.10	48.55	74.85
w/o joint training	69.30	82.44	63.35	87.89	47.25	74.16
w/o bias.supp	69.66	82.64	63.10	87.74	47.39	74.20
w/o bias.ques	69.76	82.93	63.46	88.01	47.49	74.57

Table 3: Ablation study on HotpotQA dev set. *w/o joint training* means a pipeline stepwise reasoning schema. *bias.supp* and *bias.ques* are two exposure **bias** mitigating measures to reduce the train-test discrepancy of single-hop **supp**orting sentences and **ques**tions.

and *ALBERT-xxlarge-v2* (Lan et al., 2019). From Table 4, we can see that our *StepReasoner* variants consistently perform better than the corresponding baseline models and the previous state-of-the-art method (HGN) using *ALBERT-large* as base model, especially in EM scores. It demonstrates that our method is robust to be effective based on various pre-trained models and it is the paradigm of our joint stepwise reasoning that contributes to more accurate multi-hop reasoning.

Model	Answer		Sup Fact		Joint	
	EM	F1	EM	F1	EM	F1
BERT	60.80	74.76	57.16	85.05	38.67	65.89
StepReasoner(BERT)	60.52	74.81	59.00	85.38	40.31	66.50
ELECTRA	69.49	82.76	62.80	87.91	46.75	74.33
StepReasoner(ELECTRA)	70.11	83.03	64.27	88.10	48.55	74.85
ALBERT	70.14	83.58	62.78	88.43	46.60	75.30
HGN(ALBERT)	70.18	83.44	63.17	89.19	47.01	75.74
StepReasoner(ALBERT)	70.73	83.92	64.17	88.69	48.54	75.85

Table 4: Analysis of *StepReasoner* on different backbone models on HotpotQA dev set.

Analysis of Different Reasoning Types We detailedly investigate the performance of *StepReasoner* on various reasoning types of HotpotQA compared to the baseline model. Following Min et al. (2019), HotpotQA integrates four types of multi-hop reasoning skills, including "Bridge", "Implicit-Bridge", "Comparison" and "Intersection". The first two both require identifying the bridge entity to complete the chain reasoning, but "Implicit-Bridge" resembles single-hop questions which implicitly query a multi-hop property of an entity, such as the question in Fig. 1. "Intersection" questions ask to locate the answer entity that satisfies multiple properties. "Comparison" questions involve comparing two entities to find the answer.

As shown in Table 5, our system is generally effective on all reasoning types compared to the baseline model *ELECTRA*, especially "Implicit-Bridge" and "Intersection". Because these questions are susceptible to shortcut solutions by directly identifying an entity satisfying one queried property from a single piece of evidence to reach the incorrect answer while ignoring the multi-hop reasoning involving other evidence. This observation also verifies the effectiveness of our system to stepwise generate the single-hop question grounded on the intermediate single-hop supporting sentences for interpretable multi-hop reasoning.

Reasoning Type	*ELECTRA*		*StepReasoner*	
	EM	F1	EM	F1
Bridge (34%)	47.30	76.43	48.37	76.54
Implicit-Bridge (29%)	37.04	68.66	39.81	69.65
Comparison (20%)	61.77	79.13	63.04	79.13
Intersection (17%)	42.97	73.89	46.23	75.07

Table 5: Results of Joint EM and Joint F1 across different reasoning types. The numbers in parentheses are percentages of different types.

Comparison of Different Single-hop Question Generation Methods To manifest the effectiveness of our generated single-hop questions based on identified single-hop supporting sentences, we incorporate several various single-hop question generation approaches into our stepwise reasoning framework and compare the QA results. Table 6 shows that our *Supp-based* method performs best. It reveals that our single-hop question generation is grounded on single-hop supporting sentences to generate more accurate and informative sub-questions, which are more effective than single-hop questions constructed by other strategies.

Method	Answer		Sup Fact		Joint	
	EM	F1	EM	F1	EM	F1
Span-based	68.60	81.66	62.31	87.41	46.26	73.20
USeq2Seq	69.29	82.22	63.11	88.00	46.96	74.08
Supp-based	70.11	83.03	64.27	88.10	48.55	74.85

Table 6: Comparison of various single-hop question generation methods. *Span-based* represents the sub-question generation **based** on **span** prediction in *DecompRC* (Min et al., 2019) while *USeq2Seq* is the **U**nsupervised **seq2seq** decomposition in *ONUS* (Perez et al., 2020). *Supp-based* is our **supp**orting sentences **based** single-hop question generation.

Case Study An example of the "Bridge" type question is presented in Figure 3 to show the inter-

pretable reasoning process of *StepReasoner* compared to other decomposition based methods. Our system successfully identifies the first-hop supporting sentences and generates the first-hop sub-question to query the escaping location. Then the first-hop sub-answer helps to identify the following supporting sentences to finally predict the correct answer. The second-hop question is also generated for better illustration. By contrast, *DecompRC* fails to decompose this complex question while *ONUS* generates an improper first-hop question, and they both predict a wrong answer. More cases of the other reasoning types are in Appendix B.

Question: Sparking the Marian civil war, who helped the recently abdicated queen to escape her imprisonment?
Answer: the Queen's gaoler

 # StepReasoner #

Sub-S1: The Marian civil war in Scotland (1568-1573) was a period of conflict which followed the abdication of Mary, Queen of Scots, and her escape from Loch Leven Castle in May 1568.
Sub-Q1: Where did the queen escape from during her abdication?
Sub-A1: Loch Leven Castle

Sub-S2: Loch Leven Castle is a ruined castle on an island in ... Queen of Scots was imprisoned here in 1567-1568, and forced to abdicate as queen, before escaping with the help of **her gaoler's family**.
Sub-Q2: Who helped the the queen to escape her imprisonment?

 # DecompRC #

Sub-Q: Sparking the Marian civil war, who helped the recently abdicated queen to escape her imprisonment?

 # ONUS #

Sub-Q1: Why did sparking the Marian Civil War?
Sub-Q2: Who helped the recently abdicated queen escape her imprisonment?

Wrong Predicted Answer: Loch Leven Castle

Figure 3: A case of the reasoning process by our *StepReasoner* compared to *DecompRC* and *ONUS*. The green phrase denotes our predicted answer and the texts in shadow support the single-hop question generation.

4 Related Work

Multi-hop question answering aims to aggregate multiple pieces of documents to model the multi-hop reasoning chain and predict the answer (Khashabi et al., 2018; Yang et al., 2018; Welbl et al., 2018; Ho et al., 2020; Fei et al., 2022). Prior methods mainly focus on utilizing a single reader to model the interaction between the question and relevant context, and simultaneously or separately predict the supporting sentences and answer within one step. Dhingra et al. (2018) and Qiu et al. (2019) propose to construct graphs based on entity information from scattered paragraphs and utilize graph neural networks as the reader to reason out the answer. Then HDE-Graph (Tu et al., 2019), HGN (Fang et al., 2019) and SAE (Tu et al., 2020) enrich information in the entity graph by extending nodes of other granularity to build a hierarchical graph and improve the interaction between the question and context. Some other methods (Glass et al., 2019; Beltagy et al., 2020; Zaheer et al., 2020; Alkhaldi et al., 2021) adopt pre-trained models as the powerful reader for multi-hop reasoning and achieve promising results. However, these one-step reader methods directly encode the relevant context and question for answer prediction while neglecting to illustrate the explicit reasoning process.

To circumvent the interpretability limitation, another stream of research proposes to solve the multi-hop reasoning by multi-step question decomposition. Nishida et al. (2019) and Jiang and Bansal (2019) recurrently update the sub-query at each step to break down the problem but these sub-queries are learned in latent representations and not sufficiently explainable. Instead, some works explore to explicitly decompose the complex question into single-hop questions which assumes access to decomposition supervision (Min et al., 2019; Wolfson et al., 2020). To skip this reliance, (Perez et al., 2020) and (Khot et al., 2020) attempt to construct pseudo-decomposition from other simple question corpora which pose a new challenge of label noises. Besides, they only take as input the original question to generate single-hop questions without grounding on the supporting facts at each hop. In contrast, we design a stepwise reasoning framework to locate the single-hop supporting sentences at each step for generating more fact-grounded and informative single-hop sub-questions without any genuine or pseudo supervision, and integrate the sequential reasoning process into a unified multi-hop reader for more robust performance.

5 Conclusion

In this paper, we study the task of multi-hop question answering and propose to stepwise locate the single-hop supporting sentences and generate more fact-grounded single-hop questions for better interpretable multi-hop reasoning. We present a stepwise reasoning framework to incorporate both single-hop supporting sentence identification and the corresponding single-hop question generation for each intermediate step until inferring a final result. It employs a pre-trained simple question generator and takes the identified single-hop supporting sentences as base to generate the single-hop question, which obviates the necessity of constructed supervision and helps generate more fact-based

single-hop questions. It utilizes a unified reader to jointly learn both intermediate hop reasoning and final hop inference for better fault tolerance. Experimental results validate the general effectiveness and interpretability of our *StepReasoner*.

Acknowledgments

This work is partially supported by Natural Science Foundation of China (No.6217020551, 71991471), National Social Science Foudation of China (No. 20ZDA060), Science and Technology Commission of Shanghai Municipality Grant (No.20dz1200600, 21QA1400600, 21511101000) and Zhejiang Lab (No. 2019KD0AD01).

References

Tareq Alkhaldi, Chenhui Chu, and Sadao Kurohashi. 2021. Flexibly focusing on supporting facts, using bridge links, and jointly training specialized modules for multi-hop question answering. *TASLP*.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. In *Proc. of ICLR*.

Bhuwan Dhingra, Qiao Jin, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2018. Neural models for reasoning over multiple mentions using coreference. In *Proc. of NAACL*.

Yuwei Fang, Siqi Sun, Zhe Gan, Rohit Pillai, Shuohang Wang, and Jingjing Liu. 2019. Hierarchical graph network for multi-hop question answering. In *Proc. of EMNLP*.

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.

Ruiliu Fu, Han Wang, Xuejun Zhang, Jun Zhou, and Yonghong Yan. 2021. Decomposing complex questions makes multi-hop QA easier and more interpretable. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 169–180, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ruiliu Fu, Han Wang, Jun Zhou, and Xuejun Zhang. 2022. Na-reviewer: Reviewing the context to improve the error accumulation issue for multi-hop qa. *Electronics Letters*, 58(6):237–239.

Michael Glass, Alfio Gliozzo, Rishav Chakravarti, Anthony Ferritto, Lin Pan, GP Bhargav, Dinesh Garg, and Avirup Sil. 2019. Span selection pre-training for question answering. In *Proc. of ACL*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Yichen Jiang and Mohit Bansal. 2019. Self-assembling modular networks for interpretable multi-hop reasoning. In *Proc. of EMNLP*.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proc. of NAACL*.

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. In *Proc. of NAACL*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. In *Proc. of ICLR*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proc. of ACL*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. Multi-hop reading comprehension through question decomposition and rescoring. In *Proc. of ACL*.

Kosuke Nishida, Kyosuke Nishida, Masaaki Nagata, Atsushi Otsuka, Itsumi Saito, Hisako Asano, and Junji Tomita. 2019. Answering while summarizing: Multi-task learning for multi-hop qa with evidence extraction. *arXiv preprint arXiv:1905.08511*.

Liangming Pan, Wenhu Chen, Wenhan Xiong, Min-Yen Kan, and William Yang Wang. 2020. Unsupervised multi-hop question answering by question generation. *arXiv preprint arXiv:2010.12623*.

Ethan Perez, Patrick Lewis, Wen-tau Yih, Kyunghyun Cho, and Douwe Kiela. 2020. Unsupervised question decomposition for question answering. In *Proc. of EMNLP*.

Lin Qiu, Yunxuan Xiao, Yanru Qu, Hao Zhou, Lei Li, Weinan Zhang, and Yong Yu. 2019. Dynamically fused graph network for multi-hop reasoning. *ACL*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *EMNLP*.

Nan Shao, Yiming Cui, Ting Liu, Shijin Wang, and Guoping Hu. 2020. Is graph structure necessary for multi-hop reasoning? In *Proc. of EMNLP*.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. *arXiv preprint arXiv:1803.06643*.

Ming Tu, Kevin Huang, Guangtao Wang, Jing Huang, Xiaodong He, and Bowen Zhou. 2020. Select, answer and explain: Interpretable multi-hop reading comprehension over multiple documents. In *Proc. of AAAI*.

Ming Tu, Guangtao Wang, Jing Huang, Yun Tang, Xiaodong He, and Bowen Zhou. 2019. Multi-hop reading comprehension across multiple documents by reasoning over heterogeneous graphs. In *Proc. of ACL*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *TACL*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. 2020. Break it down: A question understanding benchmark. *TACL*.

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 *Proc. of EMNLP*.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. In *Proc. of NeurIPS*.

A Training Details

All these models are implemented using Huggingface (Wolf et al., 2019). For HotpotQA, we use a batch size of 48 and fine-tune for 10 epochs with the learning rate 3e-5. For 2WikiMultiHopQA, the batch size is set to 24, the number of training epochs is 5 and the learning rate is 5e-5. The Adam is taken as the optimizer and we use a linear learning rate scheduler with 10% warmup proportion. The proposed systems and other comparison models are trained on 4 NVIDIA Tesla V100 GPUs.

B Case Study of Different Reasoning Types

We further present three cases of other reasoning types in Figure 4, including "Implicit-Bridge", "Comparison" and "Intersection". We can see that the *StepReasoner* generates high-quality decompositions for better interpretable multi-hop reasoning and predict accurate answers for all types compared to previous question decomposition based methods.

For the "Implicit-Bridge" question in Figure 4a, by first predicting the supporting sentences related to "Sivarama Swami" at the first hop, we can generate a sub-question to identify the implicit bridge "Bhaktivedanta Manor" for location query in the second hop. Although our predicted answer is different from the ground truth, it is also a reasonable response and more close to the golden one compared to the predictions of *DecompRC* and *ONUS*. These two methods both fail to decompose the multi-hop question and can only predict an intermediate answer.

For the "Comparison" and "Intersection" questions in Figure 4b and 4c, all methods predict the correct answers. However, we can generate more diverse single-hop sub-questions without requesting for any supervision, such as "die" and "death" for representing "pass away", and "belong to" for "from". By contrast, the decompositions by *DecompRC* are usually inflexibly from the original questions and the unsupervised *ONUS* creates improper sub-questions with noises. We hope that combining our generated single-hop questions can also help to construct more natural and diverse multi-hop questions and further promote multi-hop reasoning performance.

Implicit-Bridge

Question: Where does Sivarama Swami conduct courses on Vaishnava Theology?
Answer: in the village of Aldenham

 # StepReasoner #

Sub-S1: Sivarama Swami (born 30 March 1949, Budapest, Hungary) is a Vaishnava guru and a religious leader … . He has been conducting courses at Bhaktivedanta Manor on his own commentaries to … Vaishnava Theology.
Sub-Q1: Which manor does Sivarama Swami conduct courses on Vaishnava?
Sub-A1: Bhaktivedanta Manor

Sub-S2: Bhaktivedanta Manor is a Gaudiya Vaishnava temple set **in the Hertfordshire countryside of England**, in the village of Aldenham near Watford.
Sub-Q2: What section does Bhaktivedanta Manor belong to?

 # DecompRC #
Sub-Q: Where does Sivarama Swami conduct courses on Vaishnava Theology?

 # ONUS #
Sub-Q1: Where does Sivarama Swami conduct courses on Vaishnava Theology?
Sub-Q2: What is the nationality of the child?

Predicted Answer: Bhaktivedanta Manor

(a) An example of "Implicit-Bridge" reasoning type.

Comparison

Question: Who passed away first Robert Graves or Dino Buzzati?
Answer: Dino Buzzati-Traverso

 # StepReasoner #

Sub-S1: Robert von Ranke Graves (24 July 1895 - 7 December 1985), also known as Robert Ranke Graves and most commonly Robert Graves, was an English poet, novelist, critic and classicist.
Sub-Q1: When did Robert Graves die?
Sub-A1: 7 December 1985

Sub-S2: **Dino Buzzati**-Traverso (14 October 1906 - 28 January 1972) was an Italian novelist, short story writer, painter and poet, as well as a journalist … .
Sub-Q2: What was Dino Buzzati-Traverso's death date?

 # DecompRC #
Sub-Q1: Robert Graves passed away when?
Sub-Q2: Dino Buzzati passed away when? ***Sub-Q3:*** SMALLER
Predicted Answer: Dino Buzzati

 # ONUS #
Sub-Q1: When was dino buzzati born? ***Sub-Q2:*** Who passed away first?
Predicted Answer: Dino Buzzati

(b) An example of "Comparison" reasoning type.

Intersection

Question: What family are the genus' Sinofranchetia and Stauntonia from?
Answer: a genus of flowering plant in the Lardizabalaceae family

 # StepReasoner #

Sub-S1: Stauntonia is a genus of flowering plant in the Lardizabalaceae family.
Sub-Q1: What family does the Stauntonia belong to?
Sub-A1: Lardizabalaceae

Sub-S2: Sinofranchetia is a genus of flowering plant in the **Lardizabalaceae family**.
Sub-Q2: What family does the Sinofranchetia belong to?

 # DecompRC #
Sub-Q1: What family is the genus ' Sinofranchetia from?
Sub-Q2: What family is the genus ' Stauntonia from?
Sub-Q3: INTERSEC
Predicted Answer: Lardizabalaceae

 # ONUS #
Sub-Q1: What family are the genus ' Sinofranchetia?
Sub-Q2: Where is Stauntonia from?
Predicted Answer: Lardizabalaceae

(c) An example of "Intersection" reasoning type.

Figure 4: Three cases of other reasoning types. The green phrases denote our predicted answers and the texts in shadow support the corresponding single-hop question generation.