Modeling Multi-hop Question Answering as Single Sequence Prediction

Anonymous ACL submission

Abstract

Fusion-in-decoder (FID) (Izacard and Grave, 2020) is a generative question answering (QA) model that leverages passage retrieval with a pre-trained transformer and pushed the state of the art on single-hop QA. However, the complexity of multi-hop QA hinders the effectiveness of the generative QA approach. In this work, we propose a simple generative approach (PATHFID) that extends the task beyond just answer generation by explicitly modeling the reasoning process to resolve the answer for multi-hop questions. By linearizing the *hierarchical reasoning path* of supporting passages, their key sentences, and finally the factoid answer, we cast the problem as a single sequence prediction task. To facilitate complex reasoning with multiple clues, we further extend the unified flat representation of multiple input documents by encoding cross-passage interactions. Our extensive experiments demonstrate that PATHFID leads to strong performance gains on two multi-hop QA datasets: HotpotQA and IIRC. Besides the performance gains, PATHFID is more interpretable, which in turn yields answers that are more faithfully grounded to the supporting passages and facts compared to the baseline FID model.

1 Introduction

Leveraging knowledge to make complex reasoning has been a fundamental problem of artificial intelligence. Open-domain question answering (QA) (Voorhees, 1999) is an integral part of such a line of research with impactful applications (Esteva et al., 2020; Zhang et al., 2020), where the task is to answer general domain questions by gathering evidence from a large collection of documents. While super-human level performance has been achieved on single-passage reading comprehension dataset like SQuAD (Rajpurkar et al., 2016), open-domain QA still has a long way to go, especially for questions requiring more complex reasoning. The main challenge in the task of complex QA, namely *multi-hop QA*, is that it requires a QA system to combine multiple pieces of evidence from multiple documents (Welbl et al., 2018; Talmor and Berant, 2018; Yang et al., 2018). Even for single-hop QA, it has been shown challenging for extractive QA models to effectively aggregate evidence from the combined pool of multiple passages, which has been the focus of recent work (Clark and Gardner, 2018; Min et al., 2019; Guu et al., 2020).

Recent work (Lewis et al., 2020b; Min et al., 2020) has demonstrated the promise of a generative approach at combining evidences from multiple passages for answer generation. Thanks to large pre-trained transformers like T5 (Raffel et al., 2020), Izacard and Grave (2020) introduced *fusion-in-decoder* (FID) that leverages passage retrieval with generative models for open-domain QA, achieving state-of-the-art scores across several single-hop QA benchmarks. However, we observe that the success of the FID model does not extend to multi-hop QA, which is corroborated by the findings in (Xiong et al., 2021). Further, the FID model is a rather opaque model in terms of interpretation of the answer generation process. This capability becomes especially important for multi-hop QA, which requires sequential reasoning across multiple evidences from the pool of retrieved passages.

In this work, we propose PATHFID, a generative QA model that learns to generate an answer along with a *reasoning path* to improve its capability of multi-hop reasoning. PATHFID extends multi-hop QA beyond just answer generation by explicitly modeling the full reasoning path to resolve the answer with a generative sequence-to-sequence model. To this end, we cast the problem as a single sequence prediction task that simultaneously models reasoning path consisting of supporting passages and facts, and eventually the factoid answer. Furthermore, we extend PATHFID to allow for cross-passage interactions between the

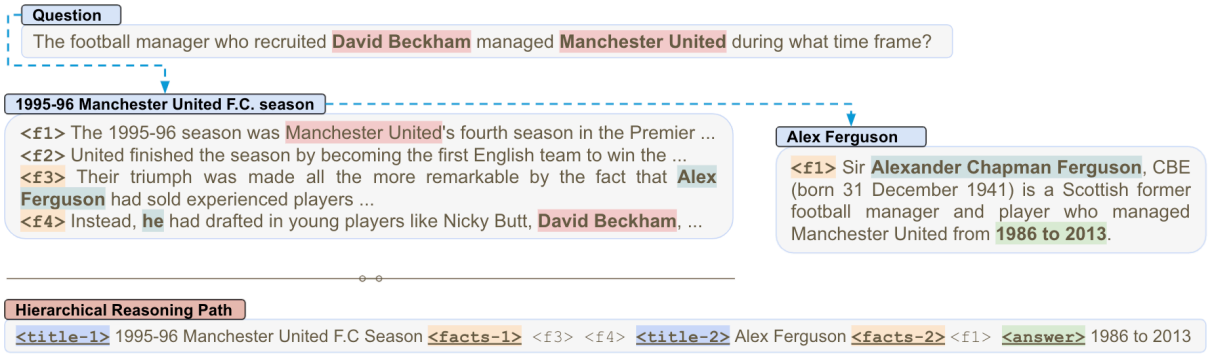

Figure 1: An example of multi-hop question from HotpotQA dataset. It requires fusing multiple evidences (supporting facts) from multiple passages in a certain order to arrive at the correct answer. We formulate the entire problem as a single sequence prediction of the linearized hierarchical path ending with the answer.

retrieved passages to obtain more expressive representations from the encoder to facilitate modeling a complex reasoning chain by the decoder. Figure 1 shows an example of our task formulation, and Figure 2 shows an overview of our approach. We evaluate our proposed approach on two multi-hop QA datasets: HotpotQA (Yang et al., 2018) and IIRC (Ferguson et al., 2020). Our extensive experiments demonstrate that (i) PATHFID leads to significant performance gains over FID on answer generation, (ii) PATHFID is the first generative model unlocking the possibility of generating the reasoning path jointly with the answer while achieving competitive performance on supporting fact extraction metric as well. Besides the performance gains, PATHFID is able to expose the underlying reasoning process behind the answer generation, which allows us to conduct a much finer-grained qualitative and quantitative analysis on the model's behavior, providing insights into further improving and better understanding generative models for multi-hop QA.

2 Problem Setup and Background

In this section, we formally introduce the problem setup and establish the necessary background.

2.1 Multi-hop Question Answering

We first describe the multi-hop QA task in a general way. We assume that a collection of K passages are given for a question q: $D_q = \{p_1, p_2, \ldots, p_K\}$, where D_q can be a pre-defined set, or it can also be an output from a text retrieval system (e.g., DPR (Karpukhin et al., 2020) and MDR (Xiong et al., 2021)) in an open-domain QA setting. That is, in the case of the open-domain setting, D_q is a subset of a large collection of passages, such as Wikipedia. The task is to generate an answer string a given q and D_q. In addition, we aim at identifying which passages provide evidence, and which sentences in them are describing the evidence. Figure 1 shows a comprehensive example of the task definition, where we can see that some sentences (called *supporting facts*) in the two paragraphs are crucial to answer the question. Moreover, there is a reasoning flow: the question $\rightarrow$ the first paragraph $\rightarrow$ the second paragraph, which is called a *reasoning path* in previous work (Asai et al., 2020). The overall task is then to predict the reasoning path along with the supporting facts, and the answer.

2.2 Fusion-in-Decoder Model (FID)

Fusion-in-Decoder (FID) is a generative reader based on a sequence-to-sequence architecture, initialized from pre-trained models such as T5 (Raffel et al., 2020) or BART (Lewis et al., 2020a). It consists of an encoder (**Enc**) and a decoder (**Dec**). First, it constructs a single block of text $b_n := \texttt{question:}\ q\ \texttt{title:}\ t_n\ \texttt{context:}\ p_n$ of concatenated evidence from each passage-title pair (p_n, t_n) together with the question (q). Then, each of the resulting evidence block b_n is independently encoded into $|b_n| \times d$-dimensional output representations, which are then concatenated to form a unified input representation

$$\mathbf{X} = [\mathbf{Enc}(b_1); \mathbf{Enc}(b_2); \ldots, \mathbf{Enc}(b_N)] \quad (1)$$

of dimension $(\sum_n |b_n|) \times d$ where $|b_n|$ denotes the length of the n-th block b_n in number of tokens. Note that, the motivation behind this strategy is to avoid the expensive quadratic self-attention computation on the encoder-side, effectively reducing the complexity from $\mathcal{O}((\sum |b_n|)^2)$ to $\mathcal{O}(\sum |b_n|^2)$. Then, the overall answer generation is modeled as a conditional generation $p_\theta(a|\mathbf{X})$ given $\mathbf{X}$ consuming the unified input representation $\mathbf{X}$, where

θ represents the set of all model parameters. The model is trained to minimize the cross-entropy loss for generating answer tokens on the decoder side. At inference time, FID first computes $\mathbf{X}$ based on the retrieved passages, and then decodes the answer token by token following $p_\theta(a_i|a_{<i}, \mathbf{X})$ with the learned model parameters θ.

3 PATHFID Reader for Multi-hop QA

In this section, we introduce a generative reader (PATHFID) for K-hop QA that jointly generates an alternating sequence of *passage-level* and *fact-level* clues on the reasoning path by more explicit fusion of evidence from the pool of input passages to arrive at the correct answer.

3.1 Overview of PATHFID

As illustrated in Figure 2, PATHFID employs a single sequence-to-sequence architecture that independently encodes the input passages after inserting special fact markers ($<f_i>$) before the i-th sentence of each passage. Conditioning on the concatenation of token-level input representations per passage, its decoder then generates the linearized hierarchical reasoning path obtained by concatenating the sequence of passage titles and their corresponding supporting fact pointers followed by the answer. Each segment on the reasoning path is separated by special markers in a way that makes it possible to uniquely recover the individual segment predictions after decoding in the inference time.

3.2 Extending Multi-hop QA beyond Answer Generation

The opaqueness of the FID model, which makes understanding of the reasoning process more difficult, motivated our approach and its emphasis on exposing the reasoning path. Instead of only modeling answer generation, we propose to jointly model it with the full *reasoning path* in an *hierarchical fashion* to derive the answer in a unified way using *multi-task maximum likelihood training*.

3.2.1 Global Input Representation

We utilize the core input encoding architecture from FID approach (Section 2.2) by introducing a new passage representation that will facilitate supporting fact generation on the reasoning path as illustrated in Figure 2. To this end, we independently encode each input passage-title pair (p_n, t_n) along with the question q as a separate block $b_n^{\text{path}} :=$ `question:` q `title:` t_n `context:` p_n^{path} where we redefine the context representation by inserting special tokens ($<f_i>$) before each sentence of the passage as

$$p_n^{\text{path}} := <f_1> s_n^{(1)} <f_2> s_n^{(2)} \cdots <f_{l_n}> s_n^{(l_n)} \quad (2)$$

where $s_n^{(i)}$ denotes the i-th sentence of passage p_n, and l_n is the number sentences it contains. Having redefined the input blocks (b_n^{path}) per passage, we then compute the global input representation similar to Eq. 1 by

$$\mathbf{X}_q^{\text{path}} = [\mathbf{Enc}(b_1^{\text{path}}); \mathbf{Enc}(b_2^{\text{path}}); \ldots; \mathbf{Enc}(b_N^{\text{path}})] \quad (3)$$

Note that sentence indicators ($<f_i>$) are shared across all passages, encouraging a more hierarchical passage representation by explicitly breaking them down into sentence-level sub-blocks using the same indicator tokens.

3.2.2 Hierarchical Reasoning Path as a Sequence

The hierarchical design of reasoning path is inspired by the human reasoning process for multi-hop QA task. More precisely, if a question q requires K-hop reasoning, then we process these K passages in a sequential order alternating between their passage-level and sentence-level evidence until we reach the answer. To this end, let $R_q = \{p_{r_1}, p_{r_2}, \ldots, p_{r_K}\}$ with $r_i \in [1, N]$ denote the sequence of passages from the larger pool D_q reflecting this reasoning process for locating the answer a for question q. As shown in Figure 2, we define the hierarchical reasoning path as a linearized sequence of blocks of passage titles and supporting facts followed by the answer block

$$\mathbf{Y}_q^{\text{path}} := [T_{r_1}; E_{r_1}; T_{r_2}; E_{r_2}; \cdots; T_K; E_{r_K}; A] \quad (4)$$

where T_{r_i} represents the i-th title block obtained by inserting a special token (`<title-i>`) before the title t_{r_j} and A denotes the answer block derived by prepending a special token (`<answer>`) to the answer a as illustrated in Figure 2. On the other hand, i-th supporting fact block is defined as the sequence of fact indicators following `<facts-i>` token by

$$E_{r_i} := \texttt{<facts-i>} <f_{j_1}> <f_{j_2}> \cdots <f_{j_{m_i}}> \quad (5)$$

where $\{j_1, j_2, \ldots, j_{m_i}\}$ denote the indices of key sentences to leverage from passage p_{r_i} to transition to the next evidence on the reasoning process

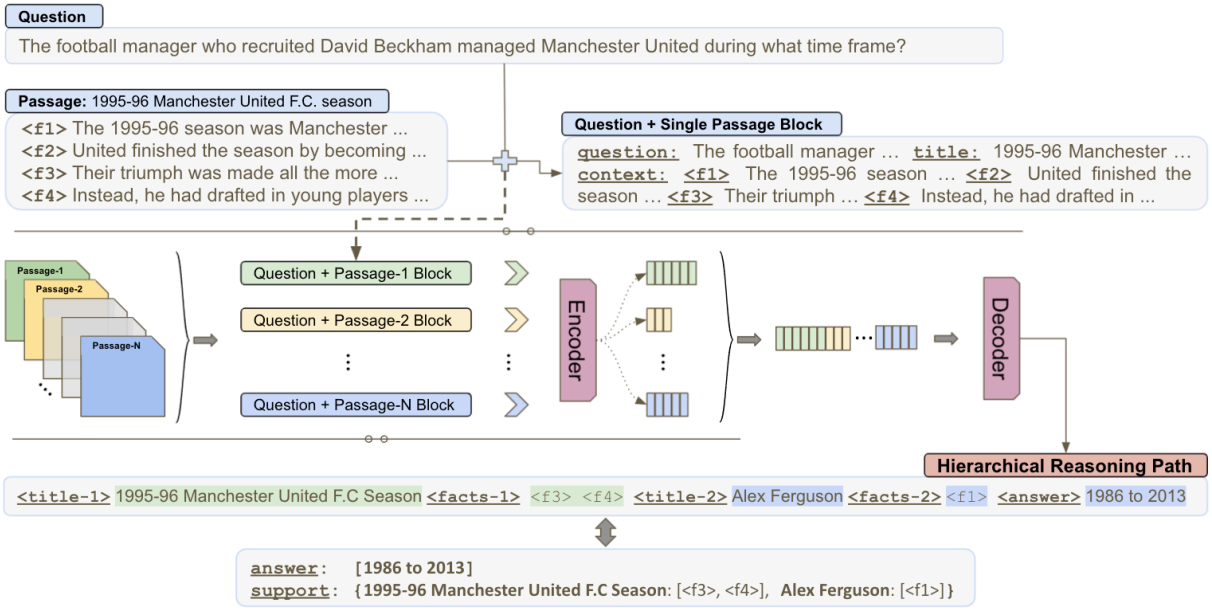

Figure 2: PATHFID model overview. Each question+passage block is encoded in parallel, which are then concatenated in to a long flat sequence of vector representations. The decoder then consumes this long sequence and generates the full reasoning path, which is then uniquely parsed into the final answer along with the supporting facts exposing the underlying reasoning.

R_q for question q, and $1 \leq m_i \leq l_{r_i}$ denotes the number of supporting facts. Note that fact indicators $<f_i>$ are shared between the contexts p_n^{path} of input blocks (Eq. 2) and supporting fact blocks (Eq. 5) on the target reasoning path to allow the decoder to follow along the sequential reasoning R_q by pointing to the facts E_{r_i} of passage p_{r_i}.

3.3 Encoding Cross-Passage Interactions (PATHFID+)

PATHFID enables more explicit evidence fusion through the reasoning path to guide the model to towards correct answer in a structured way. However, it still relies on the decoder to combine all the clues together, which might still struggle due to lack of cross-passage interactions as input blocks are encoded independently. To address this potential limitation, we propose PATHFID+, where we further extend PATHFID in a way that enables cross-passage interaction by redefining the input block consisting of a pair of passages (p_{n_1}, p_{n_2}) as

$$\begin{aligned} b^{\text{path+}}_{n_1,n_2} := \texttt{question: } q \\ \texttt{<title-1> } t_{n_1} \texttt{ <context-1> } p^{\text{path}}_{n_1} \\ \texttt{<title-2> } t_{n_2} \texttt{ <context-2> } p^{\text{path}}_{n_2} \end{aligned}$$

assuming that a set of passage pairs (p_{n_1}, p_{n_2}) are available for model to consume. In particular, we derive a set of pairs of passages from the initial set D_q by $D_q^+ = \{(p^*, p_1), (p^*, p_2), \ldots, (p^*, p_N)\}$ where p^* corresponds to the first passage that is possible to immediately hop to from question q, which may be determined by another model, or by executing the original PATHFID on D_q in our case. Global input representation $\mathbf{X}_q^{\text{path+}}$ is obtained similarly (Eq. 3) by except encoding the new blocks $b^{\text{path+}}_{n_1,n_2}$ allowing for cross-passage interactions, while the target reasoning path $\mathbf{Y}_q^{\text{path+}}$ remains the same as $\mathbf{Y}_q^{\text{path}}$. Note that <title-i> special markers are shared between new input block $b^{\text{path+}}_{n_1,n_2}$ and target reasoning path $\mathbf{Y}_q^{\text{path+}}$ to provide the model with additional clue regarding the first passage on the reasoning path while still relaying the complete evidence fusion to the decoder via information redundancy encoded in $\mathbf{X}_q^{\text{path+}}$.

3.4 Training and Inference

Having defined global input representation $\mathbf{X}_q^{\text{path}}$, the decoder autoregressively generates the reasoning path $\mathbf{Y}_q^{\text{path}}$ per token at each step by following self-attention, cross-attention on the entire $\mathbf{X}_q^{\text{path}}$, and feed-forward modules. So, the overall reasoning path generation is modeled as conditional generation $p_{\theta^{\text{path}}}(\mathbf{Y}_q^{\text{path}} | \mathbf{X}_q^{\text{path}})$. The model then is trained to minimize $J(\theta^{\text{path}}) = -\sum_{i=1}^{|\mathbf{Y}_q^{\text{path}}|} \log p_\theta(y_i | y_{<i}, \mathbf{X}_q^{\text{path}})$ with teacher forcing over a training set of $\{(q, a, D_q)\}$.

In the inference, the decoder consumes the input representation $\mathbf{X}_q^{\text{path}}$ computed by encoder, and generates the full reasoning path token by token. We then post-process the decoded sequence using

the answer indicator (<answer>) to first obtain the answer, followed by recursively parsing the remaining sequence using the special separator tokens (<title-k>, <facts-k>) to reconstruct the title and retrieve its relevant sentences at each hop k. As illustrated in Figure 2, the final result of the inference can be summarized into a dictionary which maps each generated passage title to the list of sentence pointers as well as the final answer.

4 Experiments

4.1 Datasets and General Setup

We conduct experiments on two multi-hop question answering datasets: *HotpotQA* and *IIRC*.

HotpotQA (Yang et al., 2018) is a large-scale human-annotated dataset including 113K multi-hop questions. It focuses on using documents from Wikipedia as the source of information for answering questions rather than knowledge bases as in other multi-hop QA datasets (Welbl et al., 2018; Talmor and Berant, 2018). The answer for each question in HotpotQA is extracted from 10 paragraphs in the *distractor* setting, while it is allowed to use the entire Wikipedia for the *full wiki* setting. There are two main question types *bridge* (80%) and *comparison* (20%) in the corpus. While both types require reasoning over two passages, *bridge* questions often require identifying the bridge entity in the first passage to correctly hop to the second one, which contains the answer. Each question is also provided with the annotation of 2 supporting passages and up to 5 corresponding relevant sentences as their supporting facts. Since our proposed approach is a reader model that reasons over a given set of evidence documents, we primarily focus our experiments on the *distractor* setting[1].

IIRC (Ferguson et al., 2020) is a dataset of more than 13K human-written questions over paragraphs from English Wikipedia, where crowdworkers had access only to initial paragraph and list of hyperlinks to other relevant Wikipedia articles, with the missing information occurring in one or more linked documents. This annotation design encouraged less lexical overlap between the questions and the contexts that actually contain the answer. This dataset presents unique challenges compared to HotpotQA because (1) it additionally requires discrete/numerical reasoning and identification of unanswerable questions, which adds up to 4 different possible answer types (span, binary, numerical, unanswerable), and (2) about 30% of questions require reasoning over more than 2 passages including the main passage.

Evaluation Metrics. We use standard metrics exact-match (EM) and F_1 scores for measuring the quality of predicted answers. For HotpotQA experiments, we are also able to evaluate PATHFID on supporting fact predictions using the official metrics (Support-EM, Support-F_1), which measures the performance of the reader model in correctly identifying the supporting facts from the relevant passages. Note that this metric implicitly requires correctly identifying relevant passages among the distractors as well. For our experiments on IIRC dataset, similar to the baseline model constructed in the original work (Ferguson et al., 2020), we follow the evaluation methods used by DROP (Dua et al., 2019).

Implementation Details. We use pre-trained T5-large encoder-decoder (Raffel et al., 2020) to initialize the models in our experiments. We train the model with batch size of 64 with constant learning rate of 1e-4 for 10 epochs. We use maximum length of 256 (resp. 512) tokens for input blocks of PATHFID (resp. PATHFID+), while the maximum target sequence length is set to be 64. However, the sequence truncation is performed on the reasoning path excluding answer part for sequences of length longer than 64 tokens. All the experiments are conducted on a machine with 4 or 8 many 40GB A100 GPUs. Our code is based on Huggingface Transformers (Wolf et al., 2020). Please see Appendix for further details on the hyperparameter settings.

4.2 Main Experiments: HotpotQA

4.2.1 Overall Results

We present our main results on the HotpotQA distractor setting in Table 1. We report results on the HotpotQA development set in comparison with the previous published methods. PATHFID reader provides 1.4% absolute gain on answer EM score in comparison to FID model. Moreover, it achieves competitive supporting fact predictions of 59.3% support-EM and 85.7% support-F_1 as a result of path generation compared to strong extractive models such as (Asai et al., 2020). In summary, PATHFID establishes the usefulness of modeling the full reasoning path along with answer generation for multi-hop QA. More notably, PATHFID+ achieves

[1] See Appendix C for PATHFID results in open-domain setting using MDR (Xiong et al., 2021) as the retriever.

Methods	Answer		Support	
	EM	F1	EM	F1
Baseline (Yang et al., 2018)	44.4	58.3	22.0	66.7
DFGN (Qiu et al., 2019)	55.4	69.2	-	-
QFE (Nishida et al., 2019)	53.7	68.7	58.8	84.7
SAE (Tu et al., 2020)	61.3	74.8	58.1	85.3
SAE-large (Tu et al., 2020)	67.7	80.8	63.3	87.4
Graph Recurrent Retriever (Asai et al., 2020) (base)	52.7	65.8	57.4	84.6
Graph Recurrent Retriever (Asai et al., 2020) (wwm)	68.0	81.2	58.6	85.2
Gated Memory Flow (Shao et al., 2021)	69.6	83.0	64.7	89.0
This Work				
FID* (Izacard and Grave, 2020)	64.4	77.8	-	-
PATHFID	65.8	78.9	59.3	85.7
PATHFID+	72.7	84.2	64.9	88.7

Table 1: Results on the development set of HotpotQA distractor setting in comparison with previous work. FID* indicates that the reported results are obtained by our implementation following the training details in the paper.

Criterion	FID	PATHFID	PATHFID+
Pred Answer Grounded in Gold Passages	93.9	95.3	97.7
Pred Answer Grounded in Gold Supports	90.8	92.1	95.6
Gold Answer Grounded in Pred Passages	-	96.2	98.0
Gold Answer Grounded in Pred Supports	-	95.3	97.4
Pred Answer Grounded in Pred Passages	-	96.4	97.5
Pred Answer Grounded in Pred Supports	-	90.3	94.3

Table 2: How faithfully grounded are the gold/predicted answers in gold/predicted supporting facts?

a quite significant performance gain across all the central evaluation metrics, demonstrating the importance of cross-passage interactions. Overall results validate the effectiveness of the two central modeling contributions of our proposed method. Next, we present further analysis and discussion on the unique advantages of PATHFID approach under a few central questions which motivated our research at the first place.

4.2.2 Analysis

How faithfully grounded are the generated answers on supporting facts? In Table 2, we present a detailed analysis comparing different models in terms of the faithfulness of their generated answers on both gold and predicted supporting facts. The first row focuses on the passage-level answer grounding computed by the percentage of the answers found in one of the gold supporting passages, while the second row reports the same analysis on sentence-level. We can observe that PATHFID models significantly improves on how faithfully the generated answers are grounded on the supporting facts both at passage-level and sentence-level granularities. The next two rows provide further insight into the quality of the generated supporting facts by PATHFID models by measuring how often the gold answer can be found in them. This analysis shows that the generated supporting facts are of quite high-quality including the gold answer for more than 95.3% and 96.2% at sentence-level and passage-level, respectively. The last two rows measure the faithfulness of the generated answers on the model generated supporting facts, which is not applicable to FID model as it does not perform supporting fact prediction. We observe that the generated answers are quite faithfully grounded on the predicted supporting facts, showing the path generation not only improves the answer EM performance but also successfully grounds them on the evidence it generates as part of the full reasoning path. It is important clarify that the extractive reader models can be guaranteed to output perfectly grounded answers simply by locating the answer in their predicted supporting facts. On the other hand, it is difficult for generative models to ensure 100% answer grounding simply due to its generative nature. However, we are able to provide additional evidence validating the answers generated by PATHFID are significantly grounded in the supporting facts it generates, which might implicitly indicate that the generated reasoning path tightly aligns with the model's underlying process for answer generation. Although this is a strong evidence, it is still quite implicit in exposing the model's prediction process, so we see our approach as a step in the right direction rather than a complete solution.

Performance breakdown by the number of supporting facts and question types. In Table 3, we

	Answer-EM				Support-EM			
	Comparison		Bridge		Comparison		Bridge	
# Supp Facts	FID	PATHFID	FID	PATHFID	FID	PATHFID	FID	PATHFID
2	70.4	71.8	63.3	64.6	-	86.7	-	70.0
3	66.1	68.2	62.7	63.1	-	43.4	-	30.7
4	62.2	63.8	64.3	66.5	-	5.4	-	26.2
>=5	83.3	87.5	60.0	65.0	-	0.0	-	3.8

Table 3: Performance breakdown on Answer-EM and Support-EM by question type and the number of gold supporting facts (rows). Since FID does not generate supporting facts, corresponding columns are left empty.

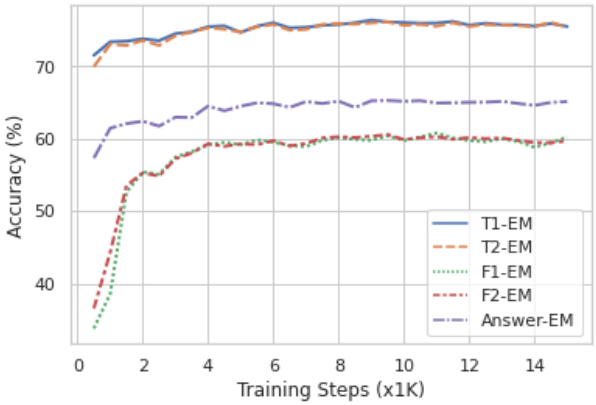

Figure 3: PATHFID model evolution on the HotpotQA Dev set during training. T1-EM, T2-EM, indicate the model's accuracy on predicting the title-1 and title-2 on the reasoning path. Similarly F1-EM, and F2-EM denote the model's accuracy on predicting set of supporting facts in passage-1 and passage-2.

compare the performance of models by breaking them down based on the number of gold supporting sentences and the question type (e.g., bridge and comparison). Our first observation is that PATHFID provides consistent improvement on answer-EM score over FID across both the question types and different number of supporting facts required to answer the question. Surprisingly, both models perform considerably well on the *comparison* questions even when it requires at least 5 supporting facts. A more important reason behind the performance breakdown analysis was to understand how the supporting fact prediction of PATHFID would change as the number of gold supporting facts grows. Although it starts degrading on examples with more than 2 supporting facts, it still achieves more than 25% Support-EM for *bridge* questions with up to 4 supporting facts. Recalling the average performance on the whole dataset is less than 60%, we conclude this result might be satisfactory enough, especially for a fully generative model on a very strict evaluation metric.

Analyzing the evolution of sub-tasks during joint training with PATHFID. In Figure 3, we present the evolution of PATHFID model on the HotpotQA development set at every 500 training steps. We observe that while the model more quickly picks up the patterns for title generation, it takes much longer for it to reach to a reasonable level of fact prediction. As one would expect, the general trend in the evolution of different segments (title-1, facts-1, title-2, facts-2, answer) of the reasoning path mostly follows the difficulty of the corresponding sub-task although all the sub-tasks are jointly formulated and trained in an end-to-end fashion. On the other hand, it seems counter-intuitive for model to reach to a better accuracy on predicting the facts of the second passage (F2-EM) on the reasoning path earlier despite having a better accuracy on (T1-EM). However, one can also interpret it as a result of stronger feedback provided by the answer segment of the reasoning path as most of the ground-truth answers are contained in the facts of the second passage.

4.3 Experiments: IIRC

In addition to our main experiments presented in greater detail, we also conduct experiments on IIRC dataset to verify the generalization of the proposed approach. To this end, we closely follow the authors' model-free retrieval setting (referred to as Oracle L+C in Table-3) because the model checkpoints for the baseline retrieval model are not available in the public release. We use a python script[2] provided in the open-sourced repository to replicate the same setting for a fair comparison.

In Table 4, we present the results on the development set for our proposed PATHFID and PATHFID+ in comparison with the baseline reported in the original paper (Ferguson et al., 2020) and our implementation of the FiD (Izacard and Grave,

[2] https://github.com/jferguson144/IIRC-baseline/blob/main/make_drop_style.py

Methods	Answer	
	EM	F1
IIRC* (Ferguson et al., 2020)	63.9	69.2
FID** (Izacard and Grave, 2020)	63.4	69.1
This Work		
PATHFID	65.2	70.5
PATHFID+	68.1	72.9

Table 4: Experimental results on IIRC dataset in model-free retrieval setting comparing the proposed method against two baselines. * indicates that the result is taken directly from the original paper (Ferguson et al., 2020) (see their Table-3), while ** indicates that we obtain the result of FID with our implementation.

2020) baseline. FID model obtains a comparable F1 with IIRC baseline with a slightly worse exact-match performance. However, the proposed PATHFID approach is able to provide 1.3% and 1.4% improvement in F1 score over the two baselines. Furthermore, PATHFID+ extension leads to the best performance achieving 4.7% and 4.2% EM score improvement in absolute value over the FID baseline and IIRC baseline, respectively. Our experimental results validate the benefit of the proposed approach on the IIRC dataset, suggesting strong evidence for the generalizability of our approach.

5 Related Work

Multi-hop question answering. Research on multi-hop QA aims to tackle complex questions that require reasoning across multiple pieces of evidence in multiple documents (Welbl et al., 2018; Yang et al., 2018; Ferguson et al., 2020). In particular, the HotpotQA dataset (Yang et al., 2018) provides both the closed and open-domain settings to evaluate multi-hop reading comprehension models. Compared to single-hop QA, such complex questions pose additional challenges; reader models (and/or retriever models) are required to capture relationships between documents, instead of independently processing each document. This is challenging because the number of document combinations exponentially grows, and even the document orders also matter as shown in Asai et al. (2020) and Xiong et al. (2021). In existing work (Xiong et al., 2019; Qi et al., 2021; Li et al., 2020; Xiong et al., 2021), most of the reading comprehension models follow a span extraction architecture (Devlin et al., 2019) with minor modifications.

Generative question answering. Especially after the emergence of the SQuAD dataset (Rajpurkar et al., 2016), neural extractive QA models have been widely studied. An underlying assumption is that we can extract a short text span (or a phrase) as an answer, but it is not always the case in reality. Motivated by this, the generative QA approach has also been investigated (Hewlett et al., 2017; Fan et al., 2019). Recent advances on pre-trained transformers have pushed this direction; for example, Lewis et al. (2020a) jointly trained a generative QA model along with a text retrieval model, and Roberts et al. (2020) explored an ambitious approach to directly generate an answer without any evidence documents. We focused on the fusion-in-decoder model (Izacard and Grave, 2020); they claimed that the decoder might be good at aggregating information across multiple documents. However, we have shown that it is not trivial in the multi-hop reasoning task, and pushed the model's ability to jointly learn to predict reasoning paths.

Open-domain question answering. Open-domain QA (Voorhees, 1999) is practically important, which requires a system to retrieve relevant documents to answer a given question. The task is recently gaining much attention, thanks to the development of large-scale datasets like HotpotQA, SQuAD Open (Chen et al., 2017), Natural Questions Open (Kwiatkowski et al., 2019; Lee et al., 2019), etc. Pre-trained transformer models like BERT (Devlin et al., 2019) have accelerated the development of neural text retrievers (Lee et al., 2019; Karpukhin et al., 2020; Asai et al., 2020; Xiong et al., 2021; Liu et al., 2021) in the retriever-reader framework (Chen et al., 2017). We have investigated the effectiveness of our method in the multi-hop open-domain QA task (see Appendix C) using an existing external retriever component.

6 Conclusion

In this work, we propose a generative question answering (QA) approach that models multi-hop QA as a single sequence prediction task. It learns to generate an answer along with a reasoning path to improve its capability of multi-hop reasoning. Our experiments on prominent multi-hop QA benchmarks, HotpotQA and IIRC, validate the promise and effectiveness of our proposed method PATHFID and its extension PATHFID+. Future work will explore (1) our PATHFID approach more closely with text retrieval models in open-domain QA scenarios and (2) more explicit grounding on the input information to make our approach even more interpretable and controllable.

References

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. 2020. Learning to retrieve reasoning paths over wikipedia graph for question answering. In *International Conference on Learning Representations*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

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)*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. 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)*.

Andre Esteva, Anuprit Kale, Romain Paulus, Kazuma Hashimoto, Wenpeng Yin, Dragomir Radev, and Richard Socher. 2020. Co-search: Covid-19 information retrieval with semantic search, question answering, and abstractive summarization. *arXiv preprint arXiv:2006.09595*.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. Eli5: Long form question answering. *arXiv preprint arXiv:1907.09190*.

James Ferguson, Matt Gardner, Hannaneh Hajishirzi, Tushar Khot, and Pradeep Dasigi. 2020. IIRC: A dataset of incomplete information reading comprehension questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, Proceedings of Machine Learning Research.

Daniel Hewlett, Llion Jones, Alexandre Lacoste, and Izzeddin Gur. 2017. Accurate supervised and semi-supervised machine reading for long documents. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Gautier Izacard and Edouard Grave. 2020. Leveraging passage retrieval with generative models for open domain question answering. *CoRR*, abs/2007.01282.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

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

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020b. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*.

Shaobo Li, Xiaoguang Li, Lifeng Shang, Xin Jiang, Qun Liu, Chengjie Sun, Zhenzhou Ji, and Bingquan Liu. 2020. Hopretriever: Retrieve hops over wikipedia to answer complex questions. *CoRR*, abs/2012.15534.

Ye Liu, Kazuma Hashimoto, Yingbo Zhou, Semih Yavuz, Caiming Xiong, and Philip Yu. 2021. Dense hierarchical retrieval for open-domain question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2021*.

Sewon Min, Danqi Chen, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019. A discrete hard EM approach for weakly supervised question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th*

International Joint Conference on Natural Language Processing (EMNLP-IJCNLP).

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. AmbigQA: Answering ambiguous open-domain questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Yixin Nie, Songhe Wang, and Mohit Bansal. 2019. Revealing the importance of semantic retrieval for machine reading at scale. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

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. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Peng Qi, Haejun Lee, Oghenetegiri "TG" Sido, and Christopher D. Manning. 2021. Retrieve, read, rerank, then iterate: Answering open-domain questions of varying reasoning steps from text.

Peng Qi, Xiaowen Lin, Leo Mehr, Zijian Wang, and Christopher D. Manning. 2019. Answering complex open-domain questions through iterative query generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Lin Qiu, Yunxuan Xiao, Yanru Qu, Hao Zhou, Lei Li, Weinan Zhang, and Yong Yu. 2019. Dynamically fused graph network for multi-hop reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

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

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*.

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

Nan Shao, Yiming Cui, Ting Liu, Shijin Wang, and Guoping Hu. 2021. Memory augmented sequential paragraph retrieval for multi-hop question answering.

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)*.

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 *AAAI 2020 (accepted)*.

Ellen M. Voorhees. 1999. The trec-8 question answering track report. In *In Proceedings of TREC-8*, pages 77–82.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.

Wenhan Xiong, Xiang Li, Srini Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Scott Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oguz. 2021. Answering complex open-domain questions with multi-hop dense retrieval. In *International Conference on Learning Representations*.

Wenhan Xiong, Mo Yu, Xiaoxiao Guo, Hong Wang, Shiyu Chang, Murray Campbell, and William Yang Wang. 2019. Simple yet effective bridge reasoning for open-domain multi-hop question answering. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*.

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

Edwin Zhang, Nikhil Gupta, Raphael Tang, Xiao Han, Ronak Pradeep, Kuang Lu, Yue Zhang, Rodrigo Nogueira, Kyunghyun Cho, Hui Fang, and Jimmy Lin. 2020. Covidex: Neural ranking models and keyword search infrastructure for the covid-19 open research dataset. *arXiv preprint arXiv:2004.05125*.

Chen Zhao, Chenyan Xiong, Corby Rosset, Xia Song, Paul Bennett, and Saurabh Tiwary. 2020. Transformer-xh: Multi-evidence reasoning with extra hop attention. In *International Conference on Learning Representations*.

A Visualizing the Correlation between Evidence and Answer

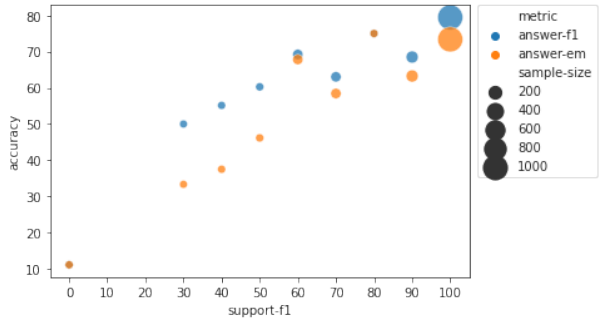

Figure 4: Visualizing the correlation between evidence and answer prediction for COMPARISON questions.

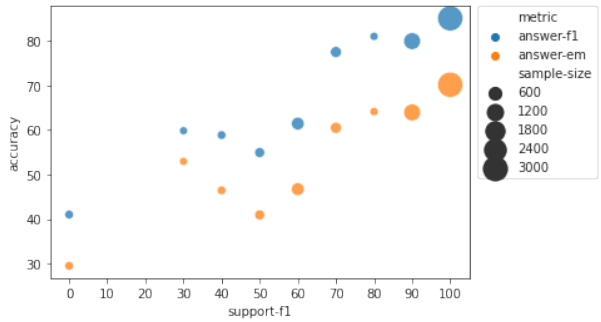

Figure 5: Visualizing the correlation between evidence and answer prediction for BRIDGE questions.

In Figure 4 and 5, we visualize the correlation between supporting evidence and answer prediction performances for *comparison* and *bridge* question types, respectively. To obtain these plots, we first split the examples into 10 buckets where n-th bucket contains the examples with support-F1 score in $(10 * (n-1), 10 * n]$ percentile for $n = \{1, 2, \ldots, 10\}$. Then, we take the average answer prediction accuracy (both EM and F1) over these examples for each bucket, and report this number on the y-axis of the plot at the corresponding support-F1 bucket on the x-axis, while dropping the empty buckets. Note that $x = 0$ corresponds to examples with support-F1 score of 0. Also note that the size of a data point on the figure reflects the number of examples in the corresponding bucket as also indicated by the legend. From Figures 4 and 5, we can observe that the accuracy of the generated answers is significantly lower, 30% for *bridge* and 10% for *comparison*, for the first bucket with zero support-F1 compared to buckets with positive support-F1 score. This suggests that the model has a difficult time figuring out the answer when the supporting evidence prediction is poor. Another observation that holds for both categories is the general trend of increased answer quality as the supporting fact prediction improves. Combining these two points provide additional evidence (in addition to Table 2 in the main paper) implicitly supporting the answer generation process of PATHFID being grounded on the generated supporting facts, which is generated as the prefix of the answer segment in the full decoded reasoning path sequence during inference.

B Analyzing the Benefit of Joint Training

In Table 5, we present the results of a case study where we analyze the benefit of multi-task training on the passage chain prediction. The first row of the table shows the results for training PATHFID only to predict the sequence of titles for the gold passages (i.e., [t1-t2]), which is just a subsequence of the full reasoning path obtained by discarding facts and the answer. The second row is another variant, where we add the answer back to the linearized target sequence while still excluding the segments corresponding to the facts. The last row correspond to the full reasoning path generation, which is corresponding to the original formulation of PATHFID as described in Section 3 and illustrated in Figure 2. Comparing first two rows in Table 5, we can immediately observe that including answer segment in the target reasoning path (i.e., [t1-t2-answer]) boosts the performance across the board although in principle it makes the task more complicated while utilizing the same underlying model capacity. Further including segments corresponding to FACTS (sentences within supporting passages) in addition to answer segment (i.e., [t1-f1-t2-f2-answer] – full reasoning path) boosts the title-EM even further, especially before applying title reconstruction post-processing step. Although the objective of the first task (i.e., [t1-t2]) is perfectly aligned with the evaluation metric used in Table 5, the performance of the resulting model remains inferior compared to jointly modeling the same task with the answer (and/or supporting facts) prediction. These two observations elicit a compelling evidence regarding the benefit of jointly modeling the sub-tasks of multi-hop QA as single sequence capturing the full reasoning path.

Reasoning Path	Generated-Title EM			Reconstructed-Title EM		
	Passage-1	Passage-2	Passage-Chain	Passage-1	Passage-2	Passage-Chain
[t1-t2]	74.3	74.8	71.6	75.4	75.4	72.9
[t1-t2-answer]	74.8	75.0	71.8	75.8	**75.8**	**73.3**
[t1-f1-t2-f2-answer]	**75.0**	**75.1**	**71.9**	**76.0**	75.6	**73.3**

Table 5: The effect of joint training as a case study on title prediction performance of PATHFID variants trained with different target reasoning paths. **Generated-Title** column corresponds to ordered passage chain prediction performance in exact-match (EM), while **Reconstructed-Title** version is computed after applying title reconstruction post-processing described in Section E.

Methods	Answer		Support	
	EM	F1	EM	F1
GoldEn Retriever (Qi et al., 2019)	-	49.8	-	64.6
Semantic Retrieval (Nie et al., 2019)	46.5	58.8	39.9	71.5
Transformer-XH (Zhao et al., 2020)	50.2	62.4	42.2	71.6
Graph Recurrent Retriever (Asai et al., 2020) (wwm)	60.5	73.3	49.3	76.1
Graph Recurrent Retriever (Asai et al., 2020) (base)	52.7	65.8	47.9	75.0
HopRetriever (Li et al., 2020)	62.1	75.2	52.5	78.9
HopRetriever-plus (Li et al., 2020)	66.6	79.2	56.0	81.8
MDR-Electra (Top-50 paths) (Xiong et al., 2021)	61.7	74.3	-	-
MDR-FiD (Top-50 paths) (Xiong et al., 2021)	61.7	73.1	-	-
Our Models				
FID* (Top-25 paths)	54.0	66.0	-	-
PATHFID (Top-25 paths)	55.8	67.9	49.0	74.1
PATHFID+ (Top-25 paths)	59.8	72.4	52.8	76.6
On Dev* Evaluation				
PATHFID+ (Top-25 paths)	70.2	81.5	60.9	86.3

Table 6: Results for open-domain setting using MDR (Xiong et al., 2021) as the retriever. Dev* refers to the development set where the retrieved passages are expanded with the gold passage (as an oracle setting) to account for the cases where the retriever fails to retrieve the gold passages. FID* indicates our implementation.

C Case Study: Full-Wiki Setting with Multi-hop Dense Retriever

In this subsection, we evaluate PATHFID in open domain setting of HotpotQA leveraging a recently proposed multi-hop dense retriever (MDR) (Xiong et al., 2021) for passage retrieval. Unlike distractor setting, MDR returns a set of passage pairs $D_q^{\text{MDR}} = \{(p_1^{(1)}, p_1^{(2)}), (p_2^{(1)}, p_2^{(2)}), \ldots, (p_N^{(1)}, p_N^{(2)})\}$ for question q, where each passage $p_n^{(i)}$ comes with a title $t_n^{(i)}$, being retrieved from Wikipedia corpus. This setting naturally fits into how we formulate PATHFID+, which operates on the pairs of input passages as introduced in Section 3.3, where we simply set $D_q^+ = D_q^{\text{MDR}}$. For experiments with FID and PATHFID, which operate on set of single input passages, we simply split the pairs into single passages, ending up with $2K$ passages when using top-K retrieved paths from MDR. We present our results for this setting in Table 6. Similar to our observation in distractor setting, PATHFID provides a significant (%1.8) answer EM score improvement over FID, while also achieving a quite competitive performance on the supporting fact prediction compared to strong discriminative models (Asai et al., 2020; Li et al., 2020) optimized for better retrieval performance. Most notably, PATHFID+ provides significant gains over PATHFID, achieving 59.8% answer-EM and 52.8% supporting fact EM score, showing the importance of encoding cross-passage interactions. It is important to note here that our results with PATHFID+ is not directly comparable to the reader results from MDR (Xiong et al., 2021) because we are able to only use top-25 retrieved paths due to hardware limitations. Finally, we also evaluate the same PATHFID+ model on Dev* obtained by

Model Size	Top-K Paths	Answer		Support	
		EM	F1	EM	F1
T5-BASE	Top-25	56.6	69.1	51.9	75.7
T5-LARGE	Top-25	59.8	72.4	52.8	76.6

Table 7: Full-wiki results with PATHFID+ comparing two different T5 model sizes.

adding the pair of gold passages in D_q^{MDR}, where we aim to isolate the error propagation from the underlying retriever. Table 6 shows that both the answer and supporting fact prediction performance improves quite significantly, showing the potential impact that developments on retriever side of the problem can also make.

D The Effect of Model Size for Future Reference

As discussed in Section E, fine-tuning PATHFID+ with T5-large initialization might require significant resources and non-trivial memory efficient optimization (e.g., gradient checkpointing). To provide a baseline with a smaller model for future research, here we include the results of PATHFID+ with T5-base initialization using the same setting reported in Table 6 in the main paper. As presented in Table 7, although the performance difference on the supporting fact prediction is relatively small (1%), answer prediction performance drops significantly (by 3.2%) when we switch from T5-large to T5-base. However, working with T5-base is much more efficient in terms of resources and iteration time for building baselines, trying out new ideas and thought experiments. So, we hope this baseline will be helpful for future research.

E More on Training and Implementation Details

Hop ordering. HotpotQA benchmark provides annotation only for *unordered* gold passages, without explicitly specifying which passage corresponds to the k-th hop (e.g., first-hop, second-hop, etc.) on the reasoning path. In our implementation, we combine the heuristic strategies applied by GRR (Asai et al., 2020) and MDR (Xiong et al., 2021). More precisely, if only one of the gold passages contains the answer, then we take the passage that includes the answer span as the final passage. If the answer span is included in both passages, we break the tie by falling back to the hyperlink-based ordering strategy proposed by GRR (Asai et al., 2020).

Post-processing for passage title reconstruction. Note that PATHFID generates the titles of the passages on the reasoning path token by token including the separator tokens. However, the decoder might fall into some minor errors during the generation process, which may cause the resulting titles to end up slightly different from the original ones. To account for such minor errors, we leverage the set of titles coming from the input passages and find the most similar among them to our generated passage titles based on token-level F1-score. We call this process *title reconstruction* and apply it while reporting the performance for supporting fact predictions. Table 5 shows the benefit of *title reconstruction* for mitigating such minor generation errors. On the other hand, the small performance boost suggests that titles PATHFID already generates quite faithful title predictions.

Model selection. For all the models reported in this work, we perform evaluation at every 500 steps during training by decoding the whole development set on a separate machine in a non-blocking fashion. We then select the best model based on the answer exact-match score performance. However, since PATHFID variants generate more than just the answer, it can be leveraged to optimize for a more holistic metric including the supporting fact prediction performance, offering further control on model selection. We leave further exploration of this phenomenon to future work.

Scaling to larger evidence pools for full-wiki setting. As briefly noted in Appendix C, we report results in full-wiki setting using only top-25 paths returned by MDR (Xiong et al., 2021) due to hardware constraints. More precisely, a single training example becomes impossible to fit into GPU memory (40GB) even for top-25 paths for PATHFID+ model with T5-large initialization. To make the training feasible, we resort to gradient checkpointing[3] which trades off GPU memory with speed. However, in this case, even with 25 retrieved paths, training PATHFID+ for 10K steps with batch size of 64 using gradient accumulation takes 19 hours on 8 A100 GPUs with 40GB memory each, which is one of the most prominent limitations hurdling the progress for this line of research. Further research on making generative approaches with large pretrained models more efficient without losing on the performance side holds a great potential impact to accelerate the progress of fully generative models

[3] https://pytorch.org/docs/stable/checkpoint.html

parameter	FID	PATHFID	PATHFID+
initialization	t5-large	t5-large	t5-large
learning rate	1e-4	1e-4	1e-4
learning rate schedule	constant	constant	constant
batch size	64	64	64
gradient checkpointing	no	no	no
maximum input length	256	256	512
maximum output length	32	64	64
warmup ratio	0	0	0
gradient clipping norm	1.0	1.0	1.0
training epoch	10	10	10
weight decay	0	0	0

Table 8: Hyperparameters for experiments on HotpotQA Distractor setting.

parameter	FID	PATHFID	PATHFID+
initialization	t5-large	t5-large	t5-large
learning rate	1e-4	1e-4	1e-4
learning rate schedule	constant	constant	constant
batch size	64	64	64
gradient checkpointing	no	no	no
maximum input length	256	256	512
maximum output length	32	64	64
warmup ratio	0	0	0
gradient clipping norm	1.0	1.0	1.0
training epoch	10	10	10
weight decay	0	0	0

Table 9: Hyperparameters for experiments on IIRC dataset.

parameter	FID	PATHFID	PATHFID+
initialization	t5-large	t5-large	t5-large
learning rate	1e-4	1e-4	1e-4
learning rate schedule	constant	constant	constant
batch size	64	64	64
gradient checkpointing	yes	yes	yes
maximum input length	256	256	512
maximum output length	32	64	64
warmup ratio	0	0	0
gradient clipping norm	1.0	1.0	1.0
training steps	10K	10K	10K
weight decay	0	0	0
top-K path retrieval	25	25	25

Table 10: Hyperparameters for experiments on HotpotQA Full-wiki setting.

for question answering.

F Hyperparameter Settings

In Tables 9, 8 and 10, we provide the full set of important hyperparameters used for the models reported both in the main paper (HotpotQA-distractor and IIRC) and in the Appendix C (HotpotQA-fullwiki), respectively.

G Qualitative Analysis

In this section, we provide examples comparing the predictions of FID and PATHFID over *bridge* and *comparison* question types. Each of the example Table 11, 12, 13 in the next pages follows a similar structure, where we include gold answer, FID answer prediction, PATHFID answer (and full path) prediction, and 5 supporting passages (out of 10) for the brevity of presentation. Among the input passages, the first two correspond to gold passages, for which we include the full content as well as highlighting the key supporting facts/sentences with orange color. The following three passages are presented as a subset of the distractors, for each of which we include a one-line content unless it plays a crucial role in distracting at least one of the models in making a wrong prediction. In this case, we also add the content of this particular passage as well as highlighting the specific distractor span/sentence causing the failure of either FID or PATHFID.

Question	The Memphis Hustle are based in a suburb of a city with a population of what in 2010?
Input Passages	1. Memphis Hustle: <f1> The Memphis Hustle are an American professional basketball team of the NBA G League announced to begin play for the 2017–18 season as an affiliate of the Memphis Grizzlies of the National Basketball Association (NBA). <f2> Based in the Memphis suburb of Southaven, Mississippi, the team will play their home games at the Landers Center. 2. Southaven, Mississippi: <f1> Southaven is a city in DeSoto County, Mississippi, United States. <f2> It is a suburb of Memphis, Tennessee, and a principal city in the Memphis metropolitan area. <f3> The 2010 census reported a population of 48,982, making Southaven the third largest city in Mississippi. <f4> Southaven is traversed from north to south by the I-55/I-69 freeway. <f5> The city's name derives from the fact that Southaven is located south of Whitehaven, a neighborhood in Memphis. 3. Lakeland, Tennessee: Lakeland is a city in Shelby County, Tennessee, and a suburb of Memphis. The population was 12,430 at the 2010 census. 4. Marion, Arkansas: Marion is a city in and the county seat of Crittenden County, Arkansas ... 5. West Memphis, Arkansas: West Memphis is the largest city in Crittenden County, Arkansas ... ...
Gold Answer	**48,982**
FID Answer	12,430
PATHFID Answer	48,982
PATHFID Output	<title-1> Memphis Hustle <facts-1> <f1> <f2> <title-2> Southaven, Mississippi <facts-2> <f1> <f2> <f3> <answer> 48,982

Table 11: BRIDGE-type question example, where **PATHFID predicts the correct answer while FID fails to do so.** The third passage is the **distractor causing FID to make a wrong prediction** due to the highlighted sentence in red.

Question	What government position was held by the woman who portrayed Corliss Archer in the film Kiss and Tell?
Input Passages	1. Kiss and Tell (1945 film): <f1> Kiss and Tell is a 1945 American comedy film starring then 17-year-old Shirley Temple as Corliss Archer. <f2> In the film, two teenage girls cause their respective parents much concern when they start to become interested in boys. <f3> The parents' bickering about which girl is the worse influence causes more problems than it solves. 2. Shirley Temple: <f1> Shirley Temple Black (April 23, 1928 – February 10, 2014) was an American actress, singer, dancer, businesswoman, and diplomat who was Hollywood's number one box-office draw as a child actress from 1935 to 1938. <f2> As an adult, she was named United States ambassador to Ghana and to Czechoslovakia and also served as Chief of Protocol of the United States. 3. Meet Corliss Archer (TV series): Meet Corliss Archer is an American television sitcom that ... 4. Meet Corliss Archer: Meet Corliss Archer, a program from radio's Golden Age, ran from ... 5. Charles Craft: Charles Craft (May 9, 1902 – September 19, 1968) was an English-born ... ...
Gold Answer	**Chief of Protocol**
FiD Answer	United States ambassador
PathFiD Answer	Chief of Protocol of the United States
PathFiD Output	<title-1> Kiss and Tell (1945 film) <facts-1> <f1> <title-2> Shirley Temple <facts-2> <f2> <answer> Chief of Protocol of the United States

Table 12: BRIDGE-type question example, where both **PathFiD and FiD fail to predict the exact gold answer.** Although the generated answers are wrong, they can both be acceptable by humans. On the other hand, both answers fail in EM accuracy, but PathFiD manages to perfectly generate the reasoning path starting from the right sentence of the correct first passage, then jumping to correct second-hop passage, followed by identifying its key sentence (<f2>), then finally locating answer in the right part of this evidence, but only failing in getting the span perfectly, which still rewards it with a reasonable F1 score. However, this example is also important in showing the possible ambiguities in questions and strictness of the exact-match accuracy metric.

Question	Which band, Letters to Cleo or Screaming Trees, had more members?
Input Passages	1. Screaming Trees: <f1> Screaming Trees was an American rock band formed in Ellensburg, Washington in 1985 by vocalist Mark Lanegan, guitarist Gary Lee Conner, bass player Van Conner and drummer Mark Pickerel. <f2> Pickerel had been replaced by Barrett Martin by the time the band reached its most successful period. <f3> Although widely associated with grunge, the band's sound incorporated hard rock and psychedelic elements. <f4> During Screaming Trees' existence the band released seven studio albums, five EPs, and three compilations. 2. Letters to Cleo: <f1> Letters to Cleo are an alternative rock band from Boston, Massachusetts, best known for the 1994 single, "Here & Now", from their full-length debut album, "Aurora Gory Alice". <f2> The band's members are Kay Hanley, Greg McKenna, Michael Eisenstein, Stacy Jones, Scott Riebling, and later, Tom Polce. 3. Change Has Come: Change Has Come was the only recording the Screaming Trees released ... 4. Jamboree (Beat Happening album): Jamboree is the second album by Beat Happening, released ... 5. Gary Lee Conner: Gary Lee Conner (born Lee Gary Conner on August 22, 1962 in Fort Irwin ... ...
Gold Answer	**Letters to Cleo**
FiD Answer	Screaming Trees
PathFiD Answer	Letters to Cleo
PathFiD Output	<title-1> Screaming Trees <facts-1> <f1> <title-2> Letters to Cleo <facts-2> <f1> <f2> <answer> Letters to Cleo

Table 13: COMPARISON-type question example, where PathFiD predicts the correct answer while FiD fails to make a correct prediction.