# MULTI-HOP QUESTION ANSWERING VIA REASONING CHAINS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Multi-hop question answering requires models to gather information from different parts of a text to answer a question. Most current approaches learn to address this task in an end-to-end way with neural networks, without maintaining an explicit representation of the reasoning process. We propose a method to extract a discrete reasoning chain over the text, which consists of a series of sentences leading to the answer. We then feed the extracted chains to a BERT-based QA model (Devlin et al., 2018) to do final answer prediction. Critically, we do not rely on gold annotated chains or "supporting facts:" at training time, we derive pseudogold reasoning chains using heuristics based on named entity recognition and coreference resolution. Nor do we rely on these annotations at test time, as our model learns to extract chains from raw text alone. We test our approach on two recently proposed large multi-hop question answering datasets: WikiHop (Welbl et al., 2018) and HotpotQA (Yang et al., 2018), and achieve state-of-art performance on WikiHop and strong performance on HotpotQA. Our analysis shows properties of chains that are crucial for high performance: in particular, modeling extraction sequentially is important, as is dealing with each candidate sentence in a context-aware way. Furthermore, human evaluation shows that our extracted chains allow humans to give answers with high confidence, indicating that these are a strong intermediate abstraction for this task.

## 1 INTRODUCTION

As high performance has been achieved in simple question answering settings (Rajpurkar et al., 2016), work on question answering has increasingly gravitated towards questions that require more complex reasoning to solve. Multi-hop question answering datasets explicitly require aggregating clues from different parts of some given documents (Dua et al., 2019; Welbl et al., 2018; Yang et al., 2018; Jansen et al., 2018; Khashabi et al., 2018). Earlier question answering datasets contain some questions of this form (Richardson et al., 2013; Lai et al., 2017), but typically exhibit a limited range of multi-hop phenomena. Designers of multi-hop datasets aim to test a range of reasoning types (Yang et al., 2018) and, ideally, systems should have to behave in a very specific way in order to do well. However, Chen & Durrett (2019) and Min et al. (2019a) show that models achieving high performance may not actually be performing the expected kinds of reasoning. Partially this is due to the difficulty of evaluating intermediate model components such as attention (Jain & Wallace, 2019), but it also suggests that models may need inductive bias if they are to solve this problem "correctly."

In this work, we propose a step in this direction, with a two-stage model that identifies intermediate *reasoning chains* and then separately determines the answer. A reasoning chain is a sequence of sentences that logically connect the question to a fact relevant (or partially relevant) to giving a reasonably supported answer. Figure 1 shows an example of what such chains look like. Extracting chains gives us a discrete intermediate output of the reasoning process, which can help us gauge our model's behavior beyond just final task accuracy. Formally, our extractor model scores sequences of sentences and produces an $n$-best list of chains via beam search.

To find the right answer, we need to maintain uncertainty over this chain set, since the correct one may not immediately be evident, and for certain types of questions, information across multiple chains may even be relevant. Sifting through the retrieved information to actually identify the answer requires deeper, more expensive computation. We employ a second-stage answer module, a BERT-

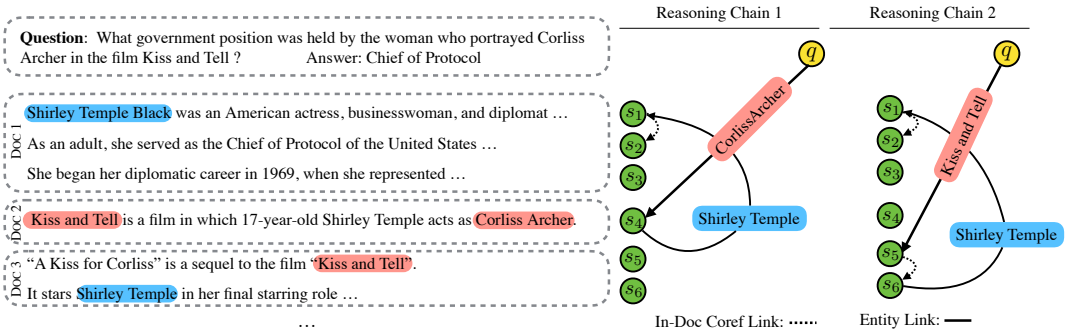


Figure 1: A multi-hop example chosen from the HotpotQA development set. Several documents are given as context to answer a question. We show two possible "reasoning chains" that leverage connections (shared entities or coreference relations) between sentences to arrive at the answer. The first chain is most appropriate, while the second requires a less well-supported inferential leap.

based QA system (Devlin et al., 2018), which can be run relatively cheaply given the pruned context. Our approach resembles past models for coarse-to-fine question answering (Choi et al., 2017; Min et al., 2018; Wang et al., 2019), but explores the context in a sequential fashion and is trained to produce principled chains.

To train our model, we heuristically label examples with reasoning chains. We use a search procedure leveraging coreference and named entity recognition (NER) to find a path from the start sentence to an end sentence through a graph of related sentences. Constructing this graph requires running an NER system at train time, but does not rely on the answer or answer candidates (Kundu et al., 2018). Our system also does not require these annotations at test time, operating instead from raw text.

Our chain identification is effective and flexible: we can use it to derive supervision on two existing datasets, and on HotpotQA (Yang et al., 2018), we found that these derived chains are essentially as effective as the ground-truth supporting fact sets labeled in the dataset. In terms of final question answering accuracy, on the WikiHop dataset (Welbl et al., 2018), our approach achieves state-of-the-art performance by a substantial margin, and on HotpotQA, we achieve strong results and outperform several recent published systems.

Our contributions are as follows: (1) We present a method for extracting oracle reasoning chains for multi-hop reasoning tasks. These chains are grounded in simple types of reasoning and generalize across multiple datasets. (2) We present a model that learns from these chains at train time and at test time can produce a list of chains. These are fed into a simple downstream model (BERT) to extract a final answer. (3) Results on two large datasets show strong performance by our chain extraction and show that our chains are intrinsically a good representation of evidence for question answering.

## 2 Question Answering via Chain Extraction

We describe our notion of chain extraction in more detail. A reasoning chain is a sequence of sentences that logically connect the question to a fact relevant to determining the answer. Two adjacent sentences in a reasoning chain should be intuitively related: they should exhibit a shared entity or event, temporal structure, or some other kind of textual relation that would allow a human reader to connect the information they contain.

Figure 1 shows an example of possible reasoning chains of an real example. In this case, we need to find information about the actor who played *Corliss Archer* in *Kiss and Tell*. These question entities may appear in multiple places in the text, and it is generally difficult to know which entity mentions might eventually lead to text containing the answer. If we arrive at $s_4$ and find the new entity *Shirley Temple*, we then need to determine what government position she held, which in this case can be found by two additional steps. Other reasoning chains could theoretically lead to this answer, such as the second chain: Shirley Temple starred in the sequel to *Kiss and Tell*, which might lead us to infer that Shirley Temple also plays Corliss Archer in Kiss and Tell. Although less justified, we also

view this as a valid reasoning chain. However, in general, there are also "connected" sequences of sentences that don't imply the answer; for example, they are connected by an entity which is not related to the question.

In determining this chain, we largely used information about entity coreference to connect the relevant pieces: either cross-document coreference about *Shirley Temple* or resolution of various pronouns. Another relevant cue is that subsequent information about *Shirley Temple* in Document 1 occurs later in the discourse, which in this case reflects temporal structure. However, solving coreference or temporal relation extraction in general is neither necessary nor sufficient to do chain extraction. Therefore, we design our system so that it does not rely on coreference at test time, but can instead directly extract reasoning chains based on what it has learned at training time.

Having established this notion of a reasoning chain, we have three questions to answer. First, how can we automatically select pseudo-ground-truth reasoning chains? Second, how do we model the chain extraction process? Third, how do we take one or more extracted chains and turn them into a final answer? We answer these three questions in the next section.

## 3 Learning to Extract Chains

### 3.1 Heuristic oracle chain construction

Following the intuition in Figure 1, we assume that there are two relevant connections between sentences that can form reasoning chains. First, the presence of a shared entity often implies some kind of connection. This is not always a sufficient clue, since common entities like *United States* may occur in otherwise unrelated sentences; however, because this oracle is only used at train time, it does not need to be 100% reliable for the model to learn a chain extraction procedure. Second, we assume that any two sentences in the same paragraph are connected; this is often true on the basis of coreference or other kinds of bridging anaphora, but these may be hard to identify automatically.

We derive heuristic reasoning chains by searching over a graph which is constructed based on these factors. Each sentence $s_i$ is represented as a node $i$ in the graph. We run an off-the-shelf named entity recognition system to extract all entities for each sentence. If sentence $i$ and sentence $j$ contain a shared entity based on string match, we add an edge between node $i$ and $j$. We then also add an edge between all pairs of sentence within the same paragraph.[1]

Starting from the question node, we do an exhaustive search to find all possible chains that could lead to the answer. This process yields a set of possible chains with different lengths; two examples are shown in Figure 1. We use two different criteria to select heuristic oracles:

- **Shortest Path**: We simply take the shortest chain from the chain set as our oracle.
- **Question Overlap**: We compute the `Rouge-F1` score for each chain's sentences with respect to the question and take the chain with the highest score. This encourages selection of more complete answer chains which address all of the question's parts without finding shortcuts.

### 3.2 Chain extraction model

Our chain extractor is a neural network that takes the input documents and questions as input and returns a variable-length sequence of sentence pointers as output.

The processing flow of our chain extractor can be divided into two main parts: sentence encoding and chain prediction as shown in Figure 2.

**Sentence Encoding** Given a document containing $n$ paragraphs and a question, we first concatenate the question with each paragraph and then encode them using the pre-trained BERT encoder (Devlin et al., 2018). We denote the encoded $i$th paragraph as $p_i$. We also encode the question by itself with BERT, denoting as $q$. To compute the representation of a sentence, we extract it from

[1]We do not explicitly run a coreference system here since current coreference systems often introduce spurious arcs. Moreover, cross-document links can nearly always be found by exact string match, and since we add all within-paragraph links, exactly determining the coreference status of every mention is not needed.

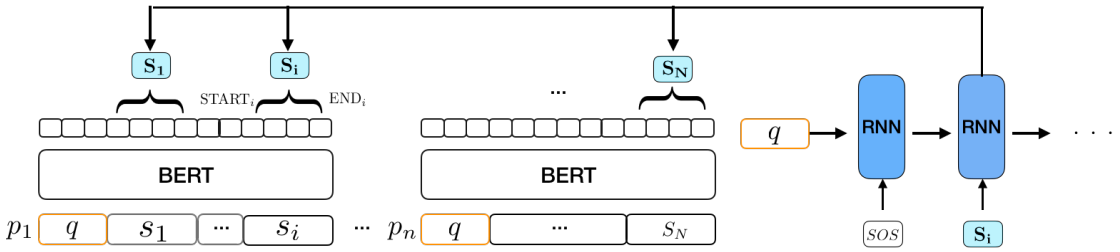


Figure 2: The BERT-Para variant of our proposed chain extractor. Left side: we encode each document paragraph jointly with the question and use pooling to form sentence representations. Right side: a pointer network extracts a sequence of sentences.

the encoded paragraph. Suppose sentence $j$ in the document is the $j$th sentence of paragraph $i$. Then $\mathbf{s}_j = \text{Span\_Extractor}(p_i, s_j^{\text{START}}, s_j^{\text{END}})$. For simplicity, we choose max-pooling as our span extractor, though other choices are possible. We name this scheme of sentence representation as `BERT-Para`. This paragraph-factored model is much more efficient and scalable than attempting to run BERT on the full context, as full contexts can be thousands of words long. We also explore an even more factored version where each sentence is concatenated with the question and encoded independently, which we denote as `BERT-Sent`. Finally, instead of using BERT as the sentence encoder, we use a bidirectional attention layer between the passage and question (Seo et al., 2017) as a baseline; we call this model `BiDAF-Para`.

**Chain Prediction** We treat all the encoded sentence representations as a bag of sentences and adopt an LSTM-based pointer network (Vinyals et al., 2015) to extract the reasoning chain, shown on the right side of Figure 2. At the first time step, we initialize the hidden state $\mathbf{h}_0$ in the pointer network using the max-pooled representation of the question $q$, and feed a special token `SOS` as the first input.

Let $c_1, \ldots, c_l$ denote the indices of sentences to include in the reasoning chain. At time step $t$, we compute the probability of sentence $i$ being chosen as $P(c_t = i | c_1, \ldots, c_{t-1}, \mathbf{s}) = \text{softmax}(\alpha)[i]$, where $\alpha_i = \mathbf{W}[\mathbf{h}_{t-1}; \mathbf{s}_{c_{t-1}}; \mathbf{h}_{t-1} \odot \mathbf{s}_{c_{t-1}}]$, and $\mathbf{W}$ is a weight matrix to be learned.

**Training the Chain Extractor** During training, the loss for time step $t$ is the negative log likelihood of the target sentence $c_t^*$ for that time step: $\text{loss}_t = -\log(P(c_t^*) | c_1^*, \ldots, c_{t-1}^* \mathbf{s})$. We also explored training with reinforcement learning. For the two datasets we considered, pre-training with our oracle and fine-tuning with policy gradient this did not lead to an improvement. Pure oracle chain extraction appears strong enough for the model to learn the needed associations across chain timesteps, but this may not be true on other datasets.

At evaluation time, we use beam search to explore a set of possible chains, which results in a set of chains $\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_k$, with each chain containing different number of sentences.

### 3.3 Answer Prediction

Since different beams may contain different plausible reasoning chains as shown in Figure 1, we consider the sentences in the top $k$ beams predicted by our chain extractor as input to our answer prediction model. Different datasets may require different modifications of the basic BERT model as well as different types of reasoning, so we present the answer prediction module in the following section.

## 4 Experimental Setup

### 4.1 Datasets

**WikiHop** Welbl et al. (2018) introduced this English dataset specially designed for text understanding across multiple documents. The dataset consists of around 40k questions, answers, and passages. Questions in this dataset are multiple-choice with around 10 choices on average.

**HotpotQA** Yang et al. (2018) proposed a new dataset with 113k English Wikipedia-based question-answer pairs. Similar to WikiHop, questions require finding and reasoning over multiple supporting documents to answer. Different from WikiHop, models should choose answers by selecting variable-length spans from these documents. Sentences relevant to finding the answer are annotated as "supporting facts" in the dataset.

### 4.2 Implementation Details

**Oracle chain extraction** We use the off-the-shelf NER system from AllenNLP (Gardner et al., 2017). We treat any entity that appears explicitly more than 5 times across sentences as a common entity,[2] and ignore it when we build the graph. Because these documents are only short snippets from Wikipedia, this criterion is loose enough to keep most useful mentions.

**Chain extractor** We use the uncased BERT tokenizer to tokenize both question and paragraphs. We use the pretrained `bert-base-uncased` model and fine-tune it using Adam with a fixed learning rate of 5e-6. At test time, we produce our chains using beam search with beam size 5.

**Answer prediction** We concatenate the question and the combined chains from previous step in the top $k$ beams in the standard way as described in the original BERT paper Devlin et al. (2018) and encode it using the pre-trained BERT model. We denote its [CLS] token as $[\text{CLS}]_p$.

WikiHop is a multiple-choice dataset. Since we need to choose an answer from a candidate list, we encode each candidate with BERT. The [CLS] token for candidate $i$ is denoted as $[\text{CLS}]_{C_i}$. We then compute the score of a candidate $C_i$ being choose as the dot product between $[\text{CLS}]_p$ and $[\text{CLS}]_{C_i}$.

HotpotQA is a span-based question answering task, where finding the answer requires predicting the start and end of a span in the context. We compute distributions over these positions via two learned weight matrices $\mathbf{W}_{\text{start}}$ and $\mathbf{W}_{\text{end}}$. Each position in the concatenated sequence except the [CLS] token is multiplied by the corresponding weight matrix and softmaxed. Since we also need to predict the question type on HotpotQA (to handle yes/no questions vs. span extraction ones), we predict the type by taking the dot product of $[\text{CLS}]_p$ with a trainable matrix $\mathbf{W}_{\text{type}}$. We use `bert-large-uncased` instead of `bert-base-uncased` in the answer prediction module. We use the same optimizer and learning rate as chain extractor.

## 5 Results

In this section we aim to answer two main questions. First, which of our proposed chain extraction techniques is most effective, and how do they compare? Second, how does our approach compare to state-of-the-art models on these datasets? Finally, can we evaluate our extracted chains intrinsically: how important is ordering and how well do they align with human intuition about question answering?

### 5.1 Comparison of Chain Extraction Methods

In this section, we study the characteristics of our extracted chains with several experiments focused on HotpotQA. We choose this dataset since it provides human-annotated supporting facts so we can directly compare these against our model.

Several statistics are shown in Table 1. For different combinations of our model and which choice of chain oracle we use, we calculate several statistics, as described in the caption. We have the following observations:

**Using more context helps chain extractors to find relevant sentences.** Comparing BERT-Para and BERT-Sent, we find that with all other parts fixed and only by encoding more context, we improve the answer prediction performance by around 5%. This may indicate that BERT can capture cross sentence relations such as coreference and find more supporting evidence as a result. The

[2]These mentions are often extremely common entities like *U.S.*, which are likely to introduce spurious edges rather than good ones.

| Model | Oracle | Avg Length | Answer Found | Supp F1 | Answer F1 |
|---|---|---|---|---|---|
| Oracle | Shortest | 1.6 | 93.6 | 58.5 | - |
| Oracle | Q-Overlap | 1.9 | 93.6 | 63.9 | - |
| Oracle | Supp Facts | 2.4 | 100.0 | 100.0 | 75.4 |
| BERT-Para | Q-Overlap | 2.0 | 76.3 | 64.5 | 66.0 |
| BERT-Para | Shortest | 1.5 | 74.1 | 56.8 | 65.5 |
| BERT-Sent | Shortest | 1.7 | 72.5 | 53.1 | 60.2 |
| BiDAF-Para | Shortest | 1.4 | 62.0 | 52.4 | 58.1 |
| BERT-Para (top 5) | Q-Overlap | 3.2 | 88.1 | 65.6 | **70.3** |

Table 1: The characteristics of different chains generated by different models under different supervision on the HotpotQA dev set: for different models and chain oracles, we report the average chain length, fraction of chains containing the answer, F1 with respect to the annotated supporting facts, and F1 on the final QA task. Here we only pick the chain in the first beam.

comparison with BiDAF-Para vs. BERT-Sent also indicates this. Despite finding many fewer answer candidates (62% instead of 72%), BiDAF-Para only achieves around 2% lower performance. One possible explanation to this is that without context, the BERT extraction model may pick up "distractor" sentences related to the question but which do not actually lead to the answer, potentially confusing the answer prediction module and cause a drop on the performance.

**The one-best chain often contains the answer.** This demonstrates the effectiveness of our chain extractor: the BERT-Para model with just 2 extracted sentences can locate the answer 76% of the time. We further analyze the quality of these chains in the following sections. Note that this is nearly the same amount of evidence as in the human-labeled supporting facts (2.4 sentences on average); the difference can be explained by cases where the model can jump directly to the answer (Chen & Durrett, 2019).

**Q-Overlap helps recover more supporting evidence.** The main difference between our Shortest oracle and the Q-Overlap oracle is that Q-Overlap contains additional relevant sentences besides the one containing the answer. As a result, models trained with Q-Overlap should also yield a higher F1 score with respect to the supporting facts, which is supported by the results (64 vs. 56).

**Performance can be improved by taking a union across multiple chains** In the last row, we show a version of BERT-Para where the top 5 chains in the beam have been unioned together and truncated to 5 sentences. These top 5 chains contain permutations of roughly the same sentences, so this does not greatly increase the average length. However, this greatly increases answer recall and downstream F1. One reason is that this maintains uncertainty over the correct reasoning chain and can seamlessly handle question types involving comparison of multiple entities, which are difficult to address with a single reasoning chain of the sort presented in Figure 1.

### 5.2 Results compared to other systems

We evaluate our best system from the prior section (BERT-Para with top-5 chains) on the blind test sets of our two datasets. Performance is shown in Table 2. On WikiHop, our model significantly outperforms past models, although these models do not use BERT. For HotpotQA, we use RoBERTa (Liu et al., 2019) weights as the pretrained model instead of BERT, which gives a performance gain. Our model also achieves strong performance compared to past models, including outperforming those which use labeled supporting facts [3]. This indicates that our heuristically-extracted chains can stand in effectively for this supervision, which suggests that our approach can generalize to settings where this annotation is not available.

### 5.3 Evaluation of chains

**Ordered extraction outperforms unordered extraction** One question we can ask is how important ordered chain extraction is versus just selecting "chain-like" sentences in an unordered fashion.

[3]Performance for other unpublished works can be find on the leader board: https://hotpotqa.github.io

| | dev | test |
|---|---|---|
| GCN (De Cao et al., 2018) | 64.8 | 67.6 |
| BAG (Cao et al., 2019) | 66.5 | 69.0 |
| CFC (Zhong et al., 2019) | 66.4 | 70.6 |
| JDReader (Tu et al., 2019) | 68.1 | 70.9 |
| DynSAN (Zhuang & Wang, 2019) | 70.1 | 71.4 |
| BERT-Para (top 5) | **72.2** | **76.5** |

| | EM | F1 | Supp? |
|---|---|---|---|
| BiDAF++ (Yang et al., 2018) | 45.60 | 59.02 | Y |
| DecompRC (Min et al., 2019b) | 55.20 | 69.63 | N |
| QFE (Nishida et al., 2019) | 53.86 | 68.06 | Y |
| DFGN (Qiu et al., 2019) | 56.31 | 69.69 | Y |
| Roberta-Para (top 5) | **61.20** | **74.11** | N |

Table 2: The blind test set performance achieved by our model on WikiHop and HotpotQA. On HotpotQA, all published works except DecompRC use the annotated supporting facts as extra supervision, which makes them not directly comparable to our model. However, our model still achieves strong performance on this dataset despite not using this annotation.

| Dataset | WikiHop | | HotpotQA | | | HotpotQA-Hard | | |
|---|---|---|---|---|---|---|---|---|
| | Acc | %ans | F1 | SP F1 | %ans | F1 | SP F1 | %ans |
| Chain Extraction | 72.4 | 72.7 | 69.7 | 63.7 | 90.3 | 56.0 | 59.2 | 78.7 |
| Unordered Extraction | 72.1 | 72.3 | 68.3 | 63.4 | 90.1 | 54.3 | 59.4 | 78.3 |

Table 3: The downstream QA performance of the chains generated by different models on different datasets. The performance is evaluated by accuracy and F1 score respectively in WikiHop and HotpotQA dataset.

We compare our BERT-Para model with a variant of our model where, instead of using a pointer network to predict a sequence, we make an independent classification decision for each sentence to determine whether it is relevant to the question or not. We then pick top $k$ sentences, for a specified value of $k$, with the highest relevance score and feed these to our BERT model. We name this model as *unordered extraction*. Both are trained with the shortest-path oracle [4]. To make a fair comparison to the unordered model, we pick the same number of sentences ranked by prediction probability as the (top-5) chain extractor.

QA performance on those datasets is shown in Table 3. Besides WikiHop and HotpotQA, we also train and test our model on a hard subset of HotpotQA pointed out by Chen & Durrett (2019). We see that **the sequential model is more powerful than the unordered model.** On all datasets, our chain extractor leads to higher QA performance than the unordered extractor, especially on HotpotQA-Hard, where multi-hop reasoning is more strongly required. This is in spite of the fact that the fraction of answers recovered is similar. This implies that even for a very powerful pre-trained model like BERT, an explicitly sequential interaction between sentences is still useful for recovering related evidences. On WikiHop, the improvement yield by our chain extractor is more marginal. One reason is that correlations have been noted between the question and answer options (Chen & Durrett, 2019), so that the quality of the extracted evidence contributes less to the models' downstream performance.

**Chain extraction is near the performance limit on HotpotQA** Given our two-stage procedure, one thing we can ask is: with a "perfect" chain extractor, how well would our question answering model do? We compare the performance of the answer prediction trained with our extracted chains against that trained with the human-annotated supporting facts. As we can see in Table 1, BERT achieves a 75.4% F1 on the annotated supporting facts, which is only 5% higher than the result achieved by our BERT-Para (top 5) extractor. A better oracle or stronger chain extractor could help close this gap, but it is already fairly small considering the headroom on the task overall. It also shows there exist other challenges to address in the question answering piece, complementary to the proposed model in this work, like decomposing the question into different pieces (Min et al., 2019b) to further improve the multi-hop QA performance.

[4]We do not use the question overlap oracle since the questions in WikiHop are synthetic like "place_of_birth gregorio di cecco", which is uninformative for the Q-overlap method.

| | quite confident | somewhat confident | not confident |
|---|---|---|---|
| shortest oracle | 34 / 77.7 | 7 / 68.6 | 9 / 70.6 |
| extracted chain | 37 / 81.1 | 7 / 64.2 | 6 / 50.0 |
| annotated supporting facts | 33 / 78.8 | 12 / 60.0 | 5 / 88.0 |

Table 4: The human evaluation on different evidence sets. For each row, 50 responses are bucketed based on the Turkers' confidence ratings, and numbers denote the answer F1 within that bucket.

**Human evaluation** We perform a human evaluation to compare the quality of our extracted chains with our oracle as well as the annotated supporting facts. We randomly pick 50 questions in HotpotQA and ask three Turkers to answer each question based on those different evidences and rate their confidence in their answer. We pick the Turkers' answer which has the highest word overlap with the actual answer (to control for Turkers who have simply misunderstood the question) and assess their confidence in it. The statistics are shown in Table 4. Human performance on the three sets is quite similar – they have similar confidence in their answers, and their answers achieve similar F1 score. Although sometimes the shortest oracle may directly jump to the answer and the extracted chains may contain distractors, humans still seem to be able to reason effectively and give confidence in their answers on these short chains. Since the difference between supporting facts and our oracle on overall question answering performance is marginal, this is further evidence that the human-annotated supporting facts may not be needed.

For examples of the chains themselves, please see Appendix A.

## 6 Related Work

**Text-based multi-hop reasoning** Memory Network based models (Weston et al., 2015; Sukhbaatar et al., 2015; Kumar et al., 2016; Dhingra et al., 2016; Shen et al., 2017) try to solve multi-hop questions sequentially by using a memory cell which is designed to gather information iteratively from different parts of the passage. These models are trained in an end-to-end fashion and the reasoning is conducted implicitly. More recent work including Entity-GCN (De Cao et al., 2018), MHQA-GRN (Song et al., 2018), and BAG (Cao et al., 2019), form this problem as a search over entity graph, and adapt graph convolution network Kipf & Welling (2017) to do reasoning. Such kind of models need to construct an entity graph both at training and test time, while we only need such entities during training.

**Coarse-to-fine question answering** Selecting the most related content regarding the question is helpful to improve the performance of a QA model. Choi et al. (2017) combine a coarse, fast model for selecting relevant sentences and a more expensive RNN for producing the answer from those sentences. Wang et al. (2019) apply distant supervision to generate labels and uses them to train a neural sentence extractor. Another line of work proposes to use the answer prediction score as supervision to the sentence extractor (Wang et al., 2018; Indurthi et al., 2018; Min et al., 2018). Instead of treating the sentence selection as a latent variable or learning the sentence extractor using policy gradient, we treat the sentence extractor as a sequence predictor and use step by step supervision generated by heuristics to train. This represents a step towards building models that represent the reasoning process more explicitly (Trivedi et al., 2019; Jiang et al., 2019).

## 7 Discussion and Conclusion

In this work, we learn to extract reasoning chains to answer multi-hop reasoning questions. Experimental results show that the chains are as effective as human annotations, and achieve strong performance on two large datasets. However, as remarked in past work (Chen & Durrett, 2019; Min et al., 2019a), there are several aspects of HotpotQA and WikiHop which make them require multi-hop reasoning less strongly than they otherwise might. As more challenging QA datasets are built based on lessons learned from these, we feel that reasoning about more explicit reasoning and properties of chain-like representations will be critical. This work represents a first step towards this goal of improving QA systems in such settings.

## REFERENCES

Yu Cao, Meng Fang, and Dacheng Tao. Bag: Bi-directional attention entity graph convolutional network for multi-hop reasoning question answering. *NAACL*, 2019.

Jifan Chen and Greg Durrett. Understanding dataset design choices for multi-hop reasoning. *NAACL*, 2019.

Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste, and Jonathan Berant. Coarse-to-fine question answering for long documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 209–220, 2017.

Nicola De Cao, Wilker Aziz, and Ivan Titov. Question answering by reasoning across documents with graph convolutional networks. *EMNLP*, 2018.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL*, 2018.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Gated-attention readers for text comprehension. *ACL*, 2016.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of NAACL-HLT*, pp. 2368–2378, 2019.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. Allennlp: A deep semantic natural language processing platform. 2017.

Satish Indurthi, Seunghak Yu, Seohyun Back, Heriberto Cuayahuitl, et al. Cut to the chase: A context zoom-in network for reading comprehension. Association for Computational Linguistics, 2018.

Sarthak Jain and Byron C. Wallace. Attention is not explanation. *NAACL*, 2019.

Peter A Jansen, Elizabeth Wainwright, Steven Marmorstein, and Clayton T Morrison. Worldtree: A corpus of explanation graphs for elementary science questions supporting multi-hop inference. *LREC*, 2018.

Yichen Jiang, Nitish Joshi, Yen-Chun Chen, and Mohit Bansal. Explore, propose, and assemble: An interpretable model for multi-hop reading comprehension. *arXiv preprint arXiv:1906.05210*, 2019.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pp. 252–262, 2018.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ICLR*, 2017.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*, pp. 1378–1387, 2016.

Souvik Kundu, Tushar Khot, and Ashish Sabharwal. Exploiting explicit paths for multi-hop reading comprehension. *arXiv preprint arXiv:1811.01127*, 2018.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale ReAding Comprehension Dataset From Examinations. *arXiv preprint arXiv:1704.04683*, 2017.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*, 2019.

Sewon Min, Victor Zhong, Richard Socher, and Caiming Xiong. Efficient and robust question answering from minimal context over documents. *ACL*, 2018.

Sewon Min, Eric Wallace, Sameer Singh, Matt Gardner, Hannaneh Hajishirzi, and Luke Zettlemoyer. Compositional questions do not necessitate multi-hop reasoning. In *ACL*, 2019a.

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. Multi-hop reading comprehension through question decomposition and rescoring. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 6097–6109, 2019b. URL https://www.aclweb.org/anthology/P19-1613/.

Kosuke Nishida, Kyosuke Nishida, Masaaki Nagata, Atsushi Otsuka, Itsumi Saito, Hisako Asano, and Junji Tomita. Answering while summarizing: Multi-task learning for multi-hop QA with evidence extraction. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 2335–2345, 2019. URL https://www.aclweb.org/anthology/P19-1225/.

Lin Qiu, Yunxuan Xiao, Yanru Qu, Hao Zhou, Lei Li, Weinan Zhang, and Yong Yu. Dynamically fused graph network for multi-hop reasoning. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 6140–6150, 2019. URL https://www.aclweb.org/anthology/P19-1617/.

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

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. MCTest: A Challenge Dataset for the Open-Domain Machine Comprehension of Text. In *EMNLP*, volume 3, pp. 4, 2013.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *ICLR*, 2017.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1047–1055. ACM, 2017.

Linfeng Song, Zhiguo Wang, Mo Yu, Yue Zhang, Radu Florian, and Daniel Gildea. Exploring Graph-structured Passage Representation for Multi-hop Reading Comprehension with Graph Neural Networks. *arXiv preprint arXiv:1809.02040*, 2018.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pp. 2440–2448, 2015.

Harsh Trivedi, Heeyoung Kwon, Tushar Khot, Ashish Sabharwal, and Niranjan Balasubramanian. Repurposing Entailment for Multi-Hop Question Answering Tasks. 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)*, 2019.

Ming Tu, Guangtao Wang, Jing Huang, Yun Tang, Xiaodong He, and Bowen Zhou. Multi-hop reading comprehension across multiple documents by reasoning over heterogeneous graphs. *arXiv preprint arXiv:1905.07374*, 2019.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, pp. 2692–2700, 2015.

Hai Wang, Dian Yu, Kai Sun, Jianshu Chen, Dong Yu, Dan Roth, and David McAllester. Evidence sentence extraction for machine reading comprehension. *arXiv preprint arXiv:1902.08852*, 2019.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerry Tesauro, Bowen Zhou, and Jing Jiang. R 3: Reinforced ranker-reader for open-domain question answering. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association of Computational Linguistics*, 6:287–302, 2018.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards AI-Complete Question Answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. *EMNLP*, 2018.

Victor Zhong, Caiming Xiong, Nitish Shirish Keskar, and Richard Socher. Coarse-grain fine-grain coattention network for multi-evidence question answering. *ICLR*, 2019.

Yimeng Zhuang and Huadong Wang. Token-level dynamic self-attention network for multi-passage reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2252–2262, 2019.

## A Appendix: Case Study of Extracted Chains

We dig into the chains picked up by our chain extractor to better understand what is captured by our model. Those examples are shown in Figure 3. Seen from example (a), the model picks a perfect chain by first picking the sentence containing "Kiss and Tell" and "Corliss Archer", then finds the next sentence through "Shirley Temple". At the last step, to our surprise, it even finds a sentence via coreference. This demonstrates that although we do not explicitly model the entity links, the model still manages to learn to jump through entities in each hop.

Example (b) shows how our system can deal with comparison-style yes/no questions. There are two entities, namely, "Laleli Mosque" and "Esma Sultan Mansion" in the question, each of which must be pursued. The chain extractor proposes first a single-sentence chain about the first entity, then a single-sentence chain about the second entity. When unioned together, our answer predictor can leverage both of these together.

Example (c) shows that the extraction model picks a sentence containing the answer but without justification, it directly jumps to the answer by the lexical overlap of the two sentences and the shared entity "South Korean". The chain picked in the second beam is a distractor. There are also different distractors that contains in other hypotheses, of which we do not put in the example. The fifth hypothesis contains the correct chain. This example shows that if the same entity appears multiple time in the document, the chain extractor may be distracted and pick unrelated distractors.

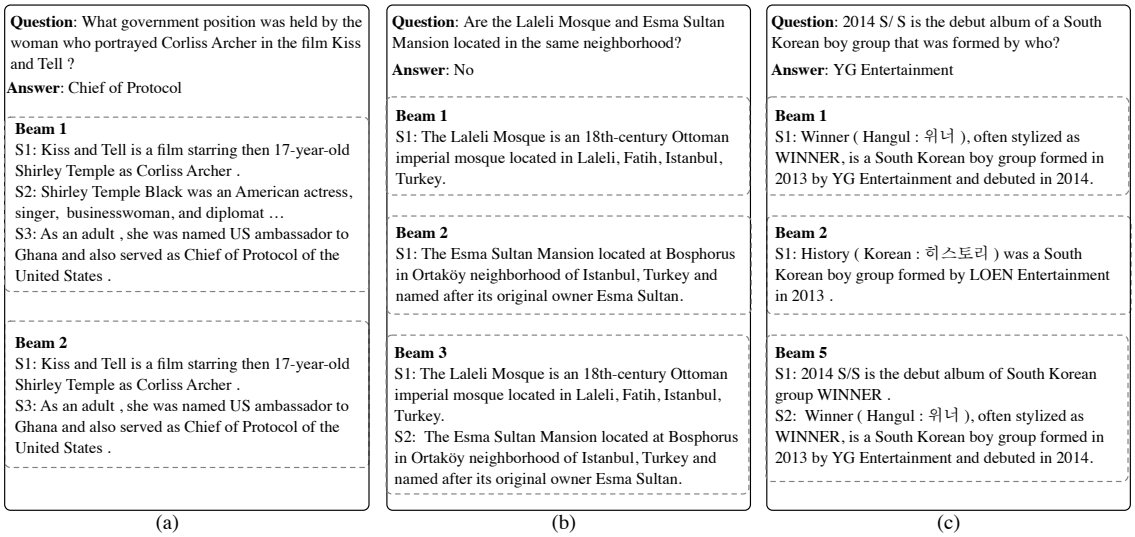


Figure 3: Examples of different chains picked up by our chain extractor on the development set of HotpotQA. The first shows a standard success case, the second shows success on a less common question type, and the third shows a failure case.