Multi-hop Reading Comprehension
through Question Decomposition and Rescoring

Sewon Min[1], Victor Zhong[1], Luke Zettlemoyer[1], Hannaneh Hajishirzi[1,2]
[1]University of Washington
[2]Allen Institute for Artificial Intelligence
{sewon,vzhong,lsz,hannaneh}@cs.washington.edu

Abstract

Multi-hop Reading Comprehension (RC) requires reasoning and aggregation across several paragraphs. We propose a system for multi-hop RC that decomposes a compositional question into simpler sub-questions that can be answered by off-the-shelf single-hop RC models. Since annotations for such decomposition are expensive, we recast sub-question generation as a span prediction problem and show that our method, trained using only 400 labeled examples, generates sub-questions that are as effective as human-authored sub-questions. We also introduce a new global rescoring approach that considers each decomposition (i.e. the sub-questions and their answers) to select the best final answer, greatly improving overall performance. Our experiments on HOTPOTQA show that this approach achieves the state-of-the-art results, while providing explainable evidence for its decision making in the form of sub-questions.

1 Introduction

Multi-hop reading comprehension (RC) is challenging because it requires the aggregation of evidence across several paragraphs to answer a question. Table 1 shows an example of multi-hop RC, where the question "Which team does the player named 2015 Diamond Head Classics MVP play for?" requires first finding the player who won MVP from one paragraph, and then finding the team that player plays for from another paragraph.

In this paper, we propose DECOMPRC, a system for multi-hop RC, that learns to break compositional multi-hop questions into simpler, single-hop sub-questions using spans from the original question. For example, for the question in Table 1, we can create the sub-questions "Which player named 2015 Diamond Head Classics MVP?" and "Which team does ANS play for?",

---

Q Which team does the player named 2015 Diamond Head Classics MVP play for?
P1 The 2015 Diamond Head Classic was ... Buddy Hield was named the tournament's MVP.
P2 Chavano Rainier Buddy Hield is a Bahamian professional basketball player for the Sacramento Kings ...

Q1 Which player named 2015 Diamond Head Classics MVP?
Q2 Which team does ANS play for?

Table 1: An example of multi-hop question from HOTPOTQA. The first cell shows given question and two of given paragraphs (other eight paragraphs are not shown), where the red text is the groundtruth answer. Our system selects a span over the question and writes two sub-questions shown in the second cell.

where the token ANS is replaced by the answer to the first sub-question. The final answer is then the answer to the second sub-question.

Recent work on question decomposition relies on distant supervision data created on top of underlying relational logical forms (Talmor and Berant, 2018), making it difficult to generalize to diverse natural language questions such as those on HOTPOTQA (Yang et al., 2018). In contrast, our method presents a new approach which simplifies the process as a span prediction, thus requiring only 400 decomposition examples to train a competitive decomposition neural model. Furthermore, we propose a rescoring approach which obtains answers from different possible decompositions and rescores each decomposition with the answer to decide on the final answer, rather than deciding on the decomposition in the beginning.

Our experiments show that DECOMPRC outperforms other published methods on HOTPOTQA (Yang et al., 2018), while providing explainable evidence in the form of sub-questions. In addition, we evaluate with alternative distractor paragraphs and questions and show that our decomposition-based approach is more ro-

bust than an end-to-end BERT baseline (Devlin et al., 2019). Finally, our ablation studies show that our sub-questions, with 400 supervised examples of decompositions, are as effective as human-written sub-questions, and that our answer-aware rescoring method significantly improves the performance.

Our code and interactive demo are publicly available at `https://github.com/shmsw25/DecompRC`.

2 Related Work

Reading Comprehension. In reading comprehension, a system reads a document and answers questions regarding the content of the document (Richardson et al., 2013). Recently, the availability of large-scale reading comprehension-datasets (Hermann et al., 2015; Rajpurkar et al., 2016; Joshi et al., 2017) has led to the development of advanced RC models (Seo et al., 2017; Xiong et al., 2018; Yu et al., 2018; Devlin et al., 2019). Most of the questions on these datasets can be answered in a single sentence (Min et al., 2018), which is a key difference from multi-hop reading comprehension.

Multi-hop Reading Comprehension. In multi-hop reading comprehension, the evidence for answering the question is scattered across multiple paragraphs. Some multi-hop datasets contain questions that are, or are based on relational queries (Welbl et al., 2017; Talmor and Berant, 2018). In contrast, HOTPOTQA (Yang et al., 2018), on which we evaluate our method, contains more natural, hand-written questions that are not based on relational queries.

Prior methods on multi-hop reading comprehension focus on answering relational queries, and emphasize attention models that reason over coreference chains (Dhingra et al., 2018; Zhong et al., 2019; Cao et al., 2019). In contrast, our method focuses on answering natural language questions via question decomposition. By providing decomposed single-hop sub-questions, our method allows the model's decisions to be explainable.

Our work is most related to Talmor and Berant (2018), which answers questions over web snippets via decomposition. There are three key differences between our method and theirs. First, they decompose questions that are correspond to relational queries, whereas we focus on natural language questions. Next, they rely on an underlying relational query (SPARQL) to build distant supervision data for training their model, while our method requires only 400 decomposition examples. Finally, they decide on a decomposition operation exclusively based on the question. In contrast, we decompose the question in multiple ways, obtain answers, and determine the best decomposition based on all given context, which we show is crucial to improving performance.

Semantic Parsing. Semantic parsing is a larger area of work that involves producing logical forms from natural language utterances, which are then usually executed over structured knowledge graphs (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Liang et al., 2011). Our work is inspired by the idea of compositionality from semantic parsing, however, we focus on answering natural language questions over unstructured text documents.

3 Model

3.1 Overview

In multi-hop reading comprehension, a system answers a question over a collection of paragraphs by combining evidence from multiple paragraphs. In contrast to single-hop reading comprehension, in which a system can obtain good performance using a single sentence (Min et al., 2018), multi-hop reading comprehension typically requires more complex reasoning over how two pieces of evidence relate to each other.

We propose DECOMPRC for multi-hop reading comprehension via question decomposition. DECOMPRC answers questions through a three step process:

1. First, DECOMPRC decomposes the original, multi-hop *question* into several single-hop *sub-questions* according to a few *reasoning types* in parallel, based on *span* predictions. Figure 1 illustrates an example in which a question is decomposed through four different reasoning types. Section 3.2 details our decomposition approach.

2. Then, for every reasoning types DECOMPRC leverages a single-hop reading comprehension model to answer each sub-question, and combines the answers according to the reasoning type. Figure 1 shows an example for which bridging produces 'City of New
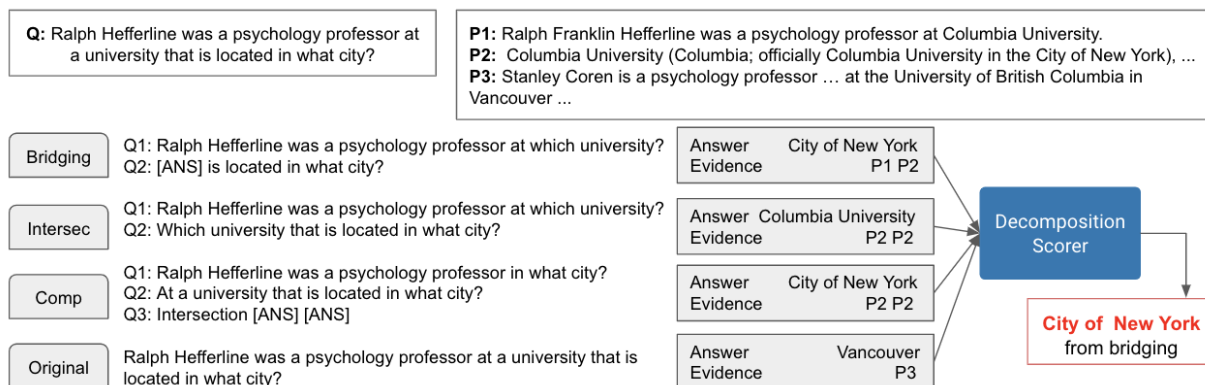
Figure 1: The overall diagram of how our system works. Given the question, DECOMPRC decomposes the question via all possible reasoning types (Section 3.2). Then, each sub-question interacts with the off-the-shelf RC model and produces the answer (Section 3.3). Lastly, the decomposition scorer decides which answer will be the final answer (Section 3.4). Here, "City of New York", obtained by bridging, is determined as a final answer.

Type	**Bridging (47%)** requires finding the first-hop evidence in order to find another, second-hop evidence.
Q	Which team does the **player** named 2015 Diamond Head Classics MVP play for?
Q1	Which player named 2015 Diamond Head Classics MVP?
Q2	Which team does ANS play for?
Type	**Intersection (23%)** requires finding an entity that satisfies two independent conditions.
Q	Stories USA starred ✓ which actor and comedian ✓ from 'The Office'?
Q1	Stories USA starred which actor and comedian?
Q2	Which actor and comedian from 'The Office'?
Type	**Comparison (22%)** requires comparing the property of two different entities.
Q	Who was born earlier, Emma Bull or Virginia Woolf?
Q1	Emma Bull was born when?
Q2	Virginia Woolf was born when?
Q3	Which is smaller (Emma Bull, ANS) (Virgina Woolf, ANS)

Table 2: The example multi-hop questions from each category of reasoning type on HOTPOTQA. Q indicates the original, multi-hop question, while Q1, Q2 and Q3 indicate sub-questions. DECOMPRC predicts span and ✓ through Pointer_c, generates sub-questions, and answers them iteratively through single-hop RC model.

York' as an answer while intersection produces 'Columbia University' as an answer. Section 3.3 details the single-hop reading comprehension procedure.

3. Finally, DECOMPRC leverages a *decomposition scorer* to judge which decomposition is the most suitable, and outputs the answer from that decomposition as the *final answer*. In Figure 1, "City of New York", obtained via bridging, is decided as the final answer. Section 3.4 details our rescoring step.

We identify several reasoning types in multi-hop reading comprehension, which we use to decompose the original question and rescore the decompositions. These reasoning types are *bridging*, *intersection* and *comparison*. Table 2 shows examples of each reasoning type. On a sample of 200 questions from the dev set of HOTPOTQA, we find that 92% of multi-hop questions belong to one of these types. Specifically, among 184 sam-

ples out of 200 which require multi-hop reasoning, 47% are bridging questions, 23% are intersection questions, 22% are comparison questions, and 8% do not belong to one of three types. In addition, these multi-hop reasoning types correspond to the types of compositional questions identified by Berant et al. (2013) and Talmor and Berant (2018).

3.2 Decomposition

The goal of question decomposition is to convert a multi-hop question into simpler, single-hop sub-questions. A key challenge of decomposition is that it is difficult to obtain annotations for how to decompose questions. Moreover, generating the question word-by-word is known to be a difficult task that requires substantial training data and is not straight-forward to evaluate (Gatt and Krahmer, 2018; Novikova et al., 2017).

Instead, we propose a method to create sub-questions using span prediction over the question.

The key idea is that, in practice, each sub-question can be formed by copying and lightly editing a key span from the original question, with different span extraction and editing required for each reasoning type. For instance, the bridging question in Table 2 requires finding "the player named 2015 Diamond Head Classic MVP" which is easily extracted as a span. Similarly, the intersection question in Table 2 specifies the type of entity to find ("which actor and comedian"), with two conditions ("Stories USA starred" and "from "The Office""), all of which can be extracted. Comparison questions compare two entities using a discrete operation over some properties of the entities, e.g., "which is smaller". When two entities are extracted as spans, the question can be converted into two sub-questions and one discrete operation over the answers of the sub-questions.

Span Prediction for Sub-question Generation
Our approach simplifies the sub-question generation problem into a span prediction problem that requires little supervision (400 annotations). The annotations are collected by mapping the question into several points that segment the question into spans (details in Section 4.2). We train a model Pointer_c that learns to map a question into c points, which are subsequently used to compose sub-questions for each reasoning type through Algorithm 1.

Pointer_c is a function that points to c indices $\text{ind}_1, \ldots, \text{ind}_c$ in an input sequence.[1] Let $S = [s_1, \ldots, s_n]$ denote a sequence of n words in the input sequence. The model encodes S using BERT (Devlin et al., 2019):

$$U = \text{BERT}(S) \in \mathbb{R}^{n \times h}, \qquad (1)$$

where h is the output dimension of the encoder.

Let $W \in \mathbb{R}^{h \times c}$ denote a trainable parameter matrix. We compute a pointer score matrix

$$Y = \text{softmax}(UW) \in \mathbb{R}^{n \times c}, \qquad (2)$$

where $\mathbb{P}(i = \text{ind}_j) = Y_{ij}$ denotes the probability that the ith word is the jth index produced by the pointer. The model extracts c indices that yield the highest joint probability at inference:

$$\text{ind}_1, \ldots, \text{ind}_c = \underset{i_1 \leq \cdots \leq i_c}{\text{argmax}} \prod_{j=1}^{c} \mathbb{P}(i_j = \text{ind}_j)$$

---

[1] c is a hyperparameter which differs in different reasoning types.

[2] Details for find_op, form_subq in Appendix B.

Algorithm 1 Sub-questions generation using Pointer_c.[2]

---
procedure GENERATESUBQ(Q : question, Pointer_c)
  /* Find q_1^b and q_2^b for Bridging */
  $\text{ind}_1, \text{ind}_2, \text{ind}_3 \leftarrow \text{Pointer}_3(Q)$
  $q_1^b \leftarrow Q_{\text{ind}_1:\text{ind}_3}$
  $q_2^b \leftarrow Q_{:\text{ind}_1} : \text{ANS} : Q_{\text{ind}_3:}$
  article in $Q_{\text{ind}_2-5:\text{ind}_2} \leftarrow$ 'which'
  /* Find q_1^i and q_2^i for Intersecion */
  $\text{ind}_1, \text{ind}_2 \leftarrow \text{Pointer}_2(Q)$
  $s_1, s_2, s_3 \leftarrow Q_{:\text{ind}_1}, Q_{\text{ind}_1:\text{ind}_2}, Q_{\text{ind}_2:}$
  if s_2 starts with wh-word **then**
    $q_1^i \leftarrow s_1 : s_2, \; q_2^i \leftarrow s_2 : s_3$
  else
    $q_1^i \leftarrow s_1 : s_2, \; q_2^i \leftarrow s_1 : s_3$
  /* Find q_1^c, q_2^c and q_3^c for Comparison */
  $\text{ind}_1, \text{ind}_2, \text{ind}_3, \text{ind}_4 \leftarrow \text{Pointer}_4(Q)$
  $\text{ent}_1, \text{ent}_2 \leftarrow Q_{\text{ind}_1:\text{ind}_2}, Q_{\text{ind}3:\text{ind}_4}$
  $op \leftarrow \text{find\_op}(Q, \text{ent}_1, \text{ent}_2)$
  $q_1^c, q_2^c \leftarrow \text{form\_subq}(Q, \text{ent}_1, \text{ent}_2, op)$
  $q_3^c \leftarrow op\,(\text{ent}_1, \text{ANS})\,(\text{ent}_2, \text{ANS})$

3.3 Single-hop Reading Comprehension

Given a decomposition, we use a single-hop RC model to answer each sub-question. Specifically, the goal is to obtain the answer and the evidence, given the sub-question and N paragraphs. Here, the answer is a span from one of paragraphs, yes or no. The evidence is one of N paragraphs on which the answer is based.

Any off-the-shelf RC model can be used. In this work, we use the BERT reading comprehension model (Devlin et al., 2019) combined with the paragraph selection approach from Clark and Gardner (2018) to handle multiple paragraphs. Given N paragraphs $S_1, \ldots, S_N$, this approach independently computes answer_i and y_i^{none} from each paragraph S_i, where answer_i and y_i^{none} denote the answer candidate from ith paragraph and the score indicating ith paragraph does not contain the answer. The final answer is selected from the paragraph with the lowest y_i^{none}. Although this approach takes a set of multiple paragraphs as an input, it is not capable of jointly reasoning across different paragraphs.

For each paragraph S_i, let $U_i \in \mathbb{R}^{n \times h}$ be the BERT encoding of the sub-question concatenated with a paragraph S_i, obtained by Equation 1. We compute four scores, y_i^{span} y_i^{yes}, y_i^{no} and y_i^{none}, indicating if the answer is a phrase in the paragraph, yes, no, or does not exist.

$$[y_i^{\text{span}}; y_i^{\text{yes}}; y_i^{\text{no}}; y_i^{\text{none}}] = \max(U_i)W_1 \in \mathbb{R}^4,$$

where max denotes a max-pooling operation across the input sequence, and $W_1 \in \mathbb{R}^{h \times 4}$ de-

notes a parameter matrix. Additionally, the model computes span_i, which is defined by its start and end points start_i and end_i.

$$\text{start}_i, \text{end}_i = \underset{j \leq k}{\operatorname{argmax}} \mathbb{P}_{i,\text{start}}(j)\mathbb{P}_{i,\text{end}}(k),$$

where $\mathbb{P}_{i,\text{start}}(j)$ and $\mathbb{P}_{i,\text{end}}(k)$ indicate the probability that the jth word is the start and the kth word is the end of the answer span, respectively. $\mathbb{P}_{i,\text{start}}(j)$ and $\mathbb{P}_{i,\text{end}}(k)$ are obtained by the jth element of p_i^{start} and the kth element of p_i^{end} from

$$p_i^{\text{start}} = \text{softmax}(U_i W_{\text{start}}) \in \mathbb{R}^n \quad (3)$$
$$p_i^{\text{end}} = \text{softmax}(U_i W_{\text{end}}) \in \mathbb{R}^n \quad (4)$$

Here, $W_{\text{start}}, W_{\text{end}} \in \mathbb{R}^h$ are the parameter matrices. Finally, answer_i is determined as one of span_i, `yes` or `no` based on which of y_i^{span}, y_i^{yes} and y_i^{no} is the highest.

The model is trained using questions that only require single-hop reasoning, obtained from SQUAD (Rajpurkar et al., 2016) and easy examples of HOTPOTQA (Yang et al., 2018) (details in Section 4.2). Once trained, it is used as an off-the-shelf RC model and is never directly trained on multi-hop questions.

3.4 Decomposition Scorer

Each decomposition consists of sub-questions, their answers, and evidence corresponding to a reasoning type. DECOMPRC scores decompositions and takes the answer of the top-scoring decomposition to be the final answer. The score indicates if a decomposition leads to a correct final answer to the multi-hop question.

Let t be the reasoning type, and let answer_t and evidence_t be the answer and the evidence from the reasoning type t. Let x denote a sequence of n words formed by the concatenation of the question, the reasoning type t, the answer answer_t, and the evidence evidence_t. The decomposition scorer encodes this input x using BERT to obtain $U_t \in \mathbb{R}^{n \times h}$ similar to Equation (1). The score p_t is computed as

$$p_t = \text{sigmoid}(W_2^T \max(U_t)) \in \mathbb{R},$$

where $W_2 \in \mathbb{R}^h$ is a trainable matrix.

During inference, the reasoning type is decided as $\operatorname{argmax}_t p_t$. The answer corresponding to this reasoning type is chosen as the final answer.

Pipeline Approach. An alternative to the decomposition scorer is a pipeline approach, in which the reasoning type is determined in the beginning, before decomposing the question and obtaining the answers to sub-questions. Section 4.6 compares our scoring step with this approach to show the effectiveness of the decomposition scorer. Here, we briefly describe the model used for the pipeline approach.

First, we form a sequence S of n words from the question and obtain $\tilde{S} \in \mathbb{R}^{n \times h}$ from Equation 1. Then, we compute 4-dimensional vector p_t by:

$$p_t = \text{softmax}(W_3 \max(\tilde{S})) \in \mathbb{R}^4$$

where $W_3 \in \mathbb{R}^{h \times 4}$ is a parameter matrix. Each element of 4-dimensional vector p_t indicates the reasoning type is bridging, intersection, comparison or original.

4 Experiments

4.1 HOTPOTQA

We experiment on HOTPOTQA (Yang et al., 2018), a recently introduced multi-hop RC dataset over Wikipedia articles. There are two types of questions—bridge and comparison. Note that their categorization is based on the data collection and is different from our categorization (bridging, intersection and comparison) which is based on the required reasoning type. We evaluate our model on dev and test sets in two different settings, following prior work.

Distractor setting contains the question and a collection of 10 paragraphs: 2 paragraphs are provided to crowd workers to write a multi-hop question, and 8 distractor paragraphs are collected separately via TF-IDF between the question and the paragraph. The train set contains easy, medium and hard examples, where easy examples are single-hop, and medium and hard examples are multi-hop. The dev and test sets are made up of only hard examples.

Full wiki setting is an open-domain setting which contains the same questions as distractor setting but does not provide the collection of paragraphs. Following Chen et al. (2017), we retrieve 30 Wikipedia paragraphs based on TF-IDF similarity between the paragraph and the question (or sub-question).

	Distractor setting					Full wiki setting				
	All	Bridge	Comp	Single	Multi	All	Bridge	Comp	Single	Multi
DECOMPRC	**70.57**	**72.53**	**62.78**	84.31	**58.74**	**43.26**	**40.30**	**55.04**	**52.11**	**35.64**
1hop train	61.73	61.57	62.36	79.38	46.53	39.17	35.30	54.57	50.03	29.83
BERT	67.08	69.41	57.81	82.98	53.38	38.40	34.77	52.85	46.14	31.74
1hop train	56.27	62.77	30.40	**87.21**	29.64	29.97	32.15	21.29	47.14	15.18
BiDAF	58.28	59.09	55.05	-	-	34.36	30.42	50.70	-	-

Table 3: F1 scores on the dev set of HOTPOTQA in both distractor (left) and full wiki settings (right). We compare DECOMPRC (our model), BERT, and BiDAF, and variants of the models that are only trained on single-hop QA data (*1hop train*). *Bridge* and *Comp* indicate original splits in HOTPOTQA; *Single* and *Multi* refer to dev set splits that can be solved (or not) by all of three BERT models trained on single-hop QA data.

Model	Dist F1	Open F1
DECOMPRC	69.63	40.65
Cognitive Graph	-	48.87
BERT Plus	69.76	-
MultiQA	-	40.23
DFGN+BERT	68.49	-
QFE	68.06	38.06
GRN	66.71	36.48
BiDAF	59.02	32.89

Table 4: F1 score on the test set of HOTPOTQA distractor and full wiki setting. All numbers from the official leaderboard. All models except BiDAF are concurrent work (not published). DECOMPRC achieves the best result out of models reported to both distractor and full wiki setting.

4.2 Implementations Details

Training Pointer for Decomposition. We obtain a set of 200 annotations for bridging to train $Pointer_3$, and another set of 200 annotations for intersection to train $Pointer_2$, hence 400 in total. Each bridging question pairs with three points in the question, and each intersection question pairs with two points in the question. For comparison, we create training data in which each question pairs with four points (the start and end of the first entity and those of the second entity) to train $Pointer_4$, requiring no extra annotation.[3]

Training Single-hop RC Model. We create *single-hop QA data* by combining HOTPOTQA easy examples and SQuAD (Rajpurkar et al., 2016) examples to form the training data for our single-hop RC model described in Section 3.3. To convert SQUAD to a multi-paragraph setting, we retrieve n other Wikipedia paragraphs based

---

[3]Details in Appendix B.

on TF-IDF similarity between the question and the paragraph, using Document Retriever from DrQA (Chen et al., 2017). We train 3 instances with $n = 0, 2, 4$ for an ensemble, which we use as the single-hop model.

To deal with ungrammatical questions generated through our decomposition procedure, we augment the training data with ungrammatical samples. Specifically, we add noise in the question by randomly dropping tokens with probability of 5%, and replace *wh*-word into 'the' with probability of 5%.

Training Decomposition Scorer We create training data by making inferences for all reasoning types on HOTPOTQA medium and hard examples. We take the reasoning type that yields the correct answer as the gold reasoning type. Appendix C provides the full details.

4.3 Baseline Models

We compare our system DECOMPRC with the state-of-the-art on the HOTPOTQA dataset as well as strong baselines.

BiDAF is the state-of-the-art RC model on HOTPOTQA, originally from Seo et al. (2017) and implemented by Yang et al. (2018).

BERT is a large, language-model-pretrained model, achieving the state-of-the-art results across many different NLP tasks (Devlin et al., 2019). This model is the same as our single-hop model described in Section 3.3, but trained on the entirety of HOTPOTQA.

BERT–1hop train is the same model but trained on single-hop QA data without HOTPOTQA medium and hard examples.

DECOMPRC–1hop train is a variant of DECOMPRC that does not use multi-hop QA data except 400 decomposition annotations. Since there

Model	F1
DECOMPRC	70.57 → 59.07
DECOMPRC–1hop train	61.73 → 58.30
BERT	67.08 → 44.68
BERT–1hop train	56.27 → 49.64

Model	Orig F1	Inv F1	Joint F1
DECOMPRC	67.80	65.78	55.80
BERT	54.65	32.49	19.27

Table 5: **Left: modifying distractor paragraphs.** F1 score on the original dev set and the new dev set made up with a different set of distractor paragraphs. DECOMPRC is our model and DECOMPRC–1hop train is DECOMPRC trained on only single-hop QA data and 400 decomposition annotations. BERT and BERT–1hop train are the baseline models, trained on HOTPOTQA and single-hop data, respectively. **Right: adversarial comparison questions.** F1 score on a subset of binary comparison questions. *Orig F1*, *Inv F1* and *Joint F1* indicate F1 score on the original example, the inverted example and the joint of two (example-wise minimum of two), respectively.

is no access to the groundtruth answers of multi-hop questions, a decomposition scorer cannot be trained. Therefore, a final answer is obtained based on the confidence score from the single-hop RC model, without a rescoring procedure.

4.4 Results

Table 3 compares the results of DECOMPRC with other baselines on the HOTPOTQA development set. We observe that DECOMPRC outperforms all baselines in both distractor and full wiki settings, outperforming the previous published result by a large margin. An interesting observation is that DECOMPRC not trained on multi-hop QA pairs (DECOMPRC–1hop train) shows reasonable performance across all data splits.

We also observe that BERT trained on single-hop RC achieves a high F1 score, even though it does not draw inferences across different paragraphs. For further analysis, we split the HOTPOTQA development set into single-hop solvable (*Single*) and single-hop non-solvable (*Multi*).[4] We observe that DECOMPRC outperforms BERT by a large margin in single-hop non-solvable (*Multi*) examples. This supports our attempt toward more explainable methods for answering multi-hop questions.

Finally, Table 4 shows the F1 score on the test set for distractor setting and full wiki setting on the leaderboard.[5] These include unpublished models that are concurrent to our work. DECOMPRC achieves the best result out of models that report both distractor and full wiki setting.

4.5 Evaluating Robustness

In order to evaluate the robustness of different methods to changes in the data distribution, we set up two adversarial settings in which the trained model remains the same but the evaluation dataset is different.

Modifying Distractor Paragraphs. We collect a new set of distractor paragraphs to evaluate if the models are robust to the change in distractors.[6] In particular, we follow the same strategy as the original approach (Yang et al., 2018) using TF-IDF similarity between the question and the paragraph, but with no overlapping distractor paragraph with the original distractor paragraphs. Table 5 compares the F1 score of DECOMPRC and BERT in the original distractor setting and in the modified distractor setting. As expected, the performance of both methods degrade, but DECOMPRC is more robust to the change in distractors. Namely, DECOMPRC–1hop train degrades much less (only 3.41 F1) compared to other approaches because it is only trained on single-hop data and therefore does not exploit the data distribution. These results confirm our hypothesis that the end-to-end model is sensitive to the change of the data and our model is more robust.

Adversarial Comparison Questions. We create an adversarial set of comparison questions by altering the original question so that the correct answer is inverted. For example, we change "Who was born earlier, Emma Bull or Virginia Woolf?" to "Who was born later, Emma Bull or Virginia Woolf?" We automatically invert 665 questions (details in Appendix D). We report the joint F1, taken as the minimum of the prediction F1 on the original and the inverted examples. Table 5 shows

---

[4]We consider an example to be solvable if all of three models of the BERT–1hop train ensemble obtains non-negative F1. This leads to 3426 single-hop solvable and 3979 single-hop non-solvable examples out of 7405 development examples, respectively.

[5]Retrieved on March 4th 2019 from `https://https://hotpotqa.github.io`

[6]We choose 8 distractor paragraphs that do not to change the groundtruth answer.

Question	Robert Smith founded the multinational company headquartered in what city?
Span-based	Q1: Robert Smith founded which multinational company? Q2: ANS headquartered in what city?
Free-form	Q1: Which multinational company was founded by Robert Smith? Q2: Which city contains a headquarter of ANS?

Table 6: An example of the original question, span-based human-annotated sub-questions and free-form human-authored sub-questions.

Sub-questions	F1
Span ($Pointer_c$ trained on 200)	65.44
Span ($Pointer_c$ trained on 400)	69.44
Span (human)	70.41
Free-form (human)	70.76

Decomposition decision method	F1
Confidence-based	61.73
Pipeline	63.59
Decomposition scorer (DECOMPRC)	70.57
Oracle	76.75

Table 7: **Left: ablations in sub-questions.** F1 score on a sample of 50 bridging questions from the dev set of HOTPOTQA, $Pointer_c$ is our span-based model trained with 200 or 400 annotations. **Right: ablations in decomposition decision method.** F1 score on the dev set of HOTPOTQA with ablating decomposition decision method. Oracle indicates that the ground truth reasoning type is selected.

the joint F1 score of DECOMPRC and BERT. We find that DECOMPRC is robust to inverted questions, and outperforms BERT by 36.53 F1.

4.6 Ablations

Span-based vs. Free-form sub-questions. We evaluate the quality of generated sub-questions using span-based question decomposition. We replace the question decomposition component using $Pointer_3$ with (i) sub-question decomposition through groundtruth spans, (ii) sub-question decomposition with free-form, hand-written sub-questions (examples shown in Table 6).

Table 7 (left) compares the question answering performance of DECOMPRC when replaced with alternative sub-questions on a sample of 50 bridging questions.[7] There is little difference in model performance between span-based and sub-questions written by human. This indicates that our span-based sub-questions are as effective as free-form sub-questions. In addition, $Pointer_3$ trained on 200 or 400 examples obtains close to human performance. We think that identifying spans often rely on syntactic information of the question, which BERT has likely learned from language modeling. We use the model trained on 200 examples for DECOMPRC to demonstrate sample-efficiency, and expect performance improvement with more annotations.

Ablations in decomposition decision method. Table 7 (right) compares different ablations to evaluate the effect of the decomposition scorer.

Breakdown of 15 failure cases	
Incorrect groundtruth	1
Partial match with the groundtruth	3
Mistake from human	3
Confusing question	1
Sub-question requires cross-paragraph reasoning	2
Decomposed sub-questions miss some information	2
Answer to the first sub-question can be multiple	3

Table 8: The error analyses of human experiment, where the upperbound F1 score of span-based sub-questions with no decomposition scorer is measured.

For comparison, we report the F1 score of the confidence-based method which chooses the decomposition with the maximum confidence score from the single-hop RC model, and the pipeline approach which independently selects the reasoning type as described in Section 3.4. In addition, we report an oracle which takes the maximum F1 score across different reasoning types to provide an upperbound. A pipeline method gets lower F1 score than the decomposition scorer. This suggests that using more context from decomposition (e.g., the answer and the evidence) helps avoid cascading errors from the pipeline. Moreover, a gap between DECOMPRC and oracle (6.2 F1) indicates that there is still room to improve.

Upperbound of Span-based Sub-questions without a decomposition scorer. To measure an upperbound of span-based sub-questions without a decomposition scorer, where a human-level RC model is assumed, we conduct a human experiment on a sample of 50 bridging ques-

---

[7]A full set of samples is shown in Appendix E.

Q What country is the Selun located in?	
P1 Selun lies between the valley of Toggenburg and Lake Walenstadt in the canton of St. Gallen.	
P2 The canton of St. Gallen is a canton of Switzerland.	

Q Which pizza chain has locations in more cities, Round Table Pizza or Marion's Piazza?	
P1 Round Table Pizza is a large chain of pizza parlors in the western United States.	
P2 Marion's Piazza ... the company currently operates 9 restaurants throughout the greater Dayton area.	
Q1 Round Table Pizza has locations in how many cities? **Q2** Marion 's Piazza has locations in how many cities?	

Q Which magazine had more previous names, Watercolor Artist or The General?	
P1 Watercolor Artist, formerly Watercolor Magic, is an American bi-monthly magazine that focuses on ...	
P2 The General (magazine): Over the years the magazine was variously called 'The Avalon Hill General', 'Avalon Hill's General', 'The General Magazine', or simply 'General'.	
Q1 Watercolor Artist had how many previous names? **Q2** The General had how many previous names?	

Table 9: The failure cases of DECOMPRC, where Q, P1 and P2 indicate the given question and paragraphs, and Q1 and Q2 indicate sub-questions from DECOMPRC. (Top) The required multi-hop reasoning is implicit, and the question cannot be decomposed. (Middle) DECOMPRC decomposes the question well but fails to answer the first sub-question because there is no explicit answer. (Bottom) DECOMPRC is incapable of counting.

tions.[8] In this experiment, humans are given each sub-question from decomposition annotations and are asked to answer it without an access to the original, multi-hop question. They are asked to answer each sub-question with no cross-paragraph reasoning, and mark it as a failure case if it is impossible. The resulting F1 score, calculated by replacing RC model to humans, is 72.67 F1.

Table 8 reports the breakdown of fifteen error cases. 53% of such cases are due to the incorrect groundtruth, partial match with the groundtruth or mistake from humans. 47% are genuine failures in the decomposition. For example, a multi-hop question "Which animal races annually for a national title as part of a post-season NCAA Division I Football Bowl Subdivision college football game?" corresponds to the last category in Table 8. The question can be decomposed into "Which post-season NCAA Division I Football Bowl Subdivision college football game?" and "Which animal races annually for a national title as part of ANS?". However in the given set of paragraphs, there are multiple games that can be the answer to the first sub-question. Although only one of them is held with the animal racing, it is impossible to get the correct answer only given the first sub-question. We think that incorporating the original question along with the sub-questions can be one solution to address this problem, which is partially done by a decomposition scorer in DECOMPRC.

Limitations. We show the overall limitations of DECOMPRC in Table 9. First, some questions are not compositional but require implicit multi-hop reasoning, hence cannot be decomposed. Sec-

ond, there are questions that can be decomposed but the answer for each sub-question does not exist explicitly in the text, and must instead by inferred with commonsense reasoning. Lastly, the required reasoning is sometimes beyond our reasoning types (e.g. counting or calculation). Addressing these remaining problems is a promising area for future work.

5 Conclusion

We proposed DECOMPRC, a system for multi-hop RC that decomposes a multi-hop question into simpler, single-hop sub-questions. We re-casted sub-question generation as a span prediction problem, allowing the model to be trained on 400 labeled examples to generate high quality sub-questions. Moreover, DECOMPRC achieved further gains from the decomposition scoring step. DECOMPRC achieved the state-of-the-art on HOTPOTQA distractor setting and full wiki setting, while providing explainable evidence for its decision making in the form of sub-questions and being more robust to adversarial settings than strong baselines.

---

[8] A full set of samples is shown in Appendix E.

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2019. Question answering by reasoning across documents with graph convolutional networks. In *NAACL*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *ACL*.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *ACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

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

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Artificial Intelligence Research*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Percy Liang, Michael Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *ACL*.

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

Jekaterina Novikova, Ondej Duek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for NLG. In *EMNLP*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch.

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

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*.

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

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *NAACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2017. Constructing Datasets for Multi-hop Reading Comprehension Across Documents. In *TACL*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2018. DCN+: Mixed objective and deep residual coattention for question answering. In *ICLR*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*.

Adams Wei Yu, David Dohan, Quoc Le, Thang Luong, Rui Zhao, and Kai Chen. 2018. Fast and accurate reading comprehension by combining self-attention and convolution. In *ICLR*.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI*.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*.

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

A Span Annotation

Figure 2: Annotation procedure. Top four figures show annotation for bridging question. Bottom three figures show annotation for intersection question.

In this section, we describe span annotation collection procedure for bridging and intersection questions.

The goal is to collect three points (bridging) or two points (intersection) given a multi-hop question. We design an interface to annotate span over the question by clicking the word in the question. First, given a question, the annotator is asked to identify which reasoning type out of bridging, intersection, one-hop and neither is the most proper.[9] Since bridging type is the most common, bridging is checked by default. If the question type is bridging, the annotator is asked to make three clicks for the start of the span, the end of

[9]Note that we exclude comparison questions for annotations, since comparison questions are already labeled on HOTPOTQA.

the span, and the head-word (top four examples in Figure 2). After three clicks are all made, the annotator can see the heuristically generated sub-questions. If the question type is intersection, the annotator is asked to make two clicks for the start and the end of the second segment out of three segments (bottom three examples in Figure 2). Similarly, the annotator can see the heuristically generated sub-questions after two clicks. If the question type is one-hop or neither, the annotator does not have to make any click. If the question can be decomposed into more than one way, the annotator is asked to choose the more natural decomposition. If the question is ambiguous, the annotator is asked to pass the example, and only annotate for the clear cases. For the quality control, all annotators have enough in person, one-on-one tutorial sessions and are given 100 example annotations for the reference.

B Decompotision for Comparison

In this section, we describe the decomposition procedure for comparison, which does not require any extra annotation.

Comparison requires to compare a property of two different entities, usually requiring discrete operations. We identify 10 discrete operations which sufficently cover comparison operations, shown in Table 10. Based on these pre-defined discrete operations, we decompose the question through the following three steps.

First, we extract two entities under comparison. We use Pointer_4 to obtain $\text{ind}_1, \ldots, \text{ind}_4$, where ind_1 and ind_2 indicate the start and the end of the first entity, and ind_3 and ind_4 indicate those of the second entity. We create a training data which each example contains the question and four points as follows: we filter out bridge questions in HOTPOTQA to leave comparison questions, extract the entities using Spacy[10] NER tagger in the question and in two supporting facts (annotated sentences in the dataset which serve as evidence to answer the question), and match them to find two entities which appear in one supporting sentence but not in the other supporting sentence.

Then, we identity the suitable discrete operation, following Algorithm 2.

Finally, we generate sub-questions according to the discrete operation. Two sub-questions are obtained for each entity.

[10]https://spacy.io/

Operation & Example
Type: Numeric Is greater (ANS) (ANS) → `yes` or `no` Is smaller (ANS) (ANS) → `yes` or `no` Which is greater (ENT, ANS) (ENT, ANS) → ENT Which is smaller (ENT, ANS) (ENT, ANS) → ENT

Did <mark>the Battle of Stones River</mark> occur before <mark>the Battle of Saipan</mark>?
    Q1: The Battle of Stones River occur when? → 1862
    Q2: The Battle of Saipan River occur when? → 1944
    Q3: Is smaller (the Battle of Stones River, 1862) (the Battle of Saipan, 1944) → `yes`

Type: Logical And (ANS) (ANS) → `yes` or `no` Or (ANS) (ANS) → `yes` or `no` Which is true (ENT, ANS) (ENT, ANS) → ENT

In between <mark>Atsushi Ogata</mark> and <mark>Ralpha Smart</mark> who graduated from Harvard College?
    Q1: Atsushi Ogata graduated from Harvard College? → `yes`
    Q2: Ralpha Smart graduated from Harvard College? → `no`
    Q3: Which is true (Atsushi Ogata, `yes`) (Ralpha Smart, `no`) → Atsushi Ogata

Type: String Is equal (ANS) (ANS) → `yes` or `no` Not equal (ANS) (ANS) → `yes` or `no` Intersection (ANS) (ANS) → string

Are <mark>Cardinal Health</mark> and <mark>Kansas City Southern</mark> located in the same state?
    Q1: Cardinal Health located in which state? → Ohio
    Q2: Cardinal Health located in which state? → Missouri
    Q3: Is equal (Ohio) (Missouri) → `no`

Table 10: A set of discrete operations proposed for comparison questions, along with the example on each type. ANS is the answer of each query, and ENT is the entity corresponding to each query. The answer of each query is shown in the right side of →. If the question and two entities for comparison are given, queries and a discrete operation can be obtained by heuristics.

C   Implementation Details

Implementation Details. We use PyTorch (Paszke et al., 2017) on top of Hugging Face's BERT implementation.[11] We tune our model from Google's pretrained BERT-BASE (lowercased)[12], containing 12 layers of Transformers (Vaswani et al., 2017) and a hidden dimension of 768. We optimize the objective function using Adam (Kingma and Ba, 2015) with learning rate 5×10^{-5}. We lowercase the input and set the maximum sequence length $|S|$ to 300 for models which input is both the question and the paragraph, and 50 for the models which input is the question only.

D   Creating Inverted Binary Comparison Questions

We identify the comparison question with 7 out of 10 discrete operations (Is greater, Is smaller,

Which is greater, Which is smaller, Which is true, Is equal, Not equal) can automatically be inverted. It leads to 665 inverted questions.

E   A Set of Samples used for Ablations

A set of samples used for ablations in Section 4.6 is shown in Table 11.

---

[11] https://github.com/huggingface/pytorch-pretrained-BERT
[12] https://github.com/google-research/bert

Algorithm 2 Algorithm for Identifying Discrete Operation. First, given two entities for comparison, the coordination and the preconjunct or the predeterminer are identified. Then, the quantitative indicator and the head entity is identified if they exist, where a set of uantitative indicators is pre-defined. In case any quantitative indicator exists, the discrete operation is determined as one of numeric operations. If there is no quantitative indicator, the discrete operation is determined as one of logical operations or string operations.

---

procedure FIND_OPERATION(question, entity1, entity2)
    coordination, preconjunct $\leftarrow f$(question, entity1, entity2)
    Determine if the question is *either* question or *both* question from coordination and preconjunct
    head entity $\leftarrow f_{head}$(question, entity1, entity2)
    if *more, most, later, last, latest, longer, larger, younger, newer, taller, higher* in question **then**
        if head entity exists **then** discrete_operation $\leftarrow$ Which is greater
        else discrete_operation $\leftarrow$ Is greater
    else if *less, earlier, earliest, first, shorter, smaller, older, closer* in question **then**
        if head entity exists **then** discrete_operation $\leftarrow$ Which is smaller
        else discrete_operation $\leftarrow$ Is smaller
    else if head entity exists **then**
        discrete_operation $\leftarrow$ Which is true
    else if question is not yes/no question and asks for the property in common **then**
        discrete_operation $\leftarrow$ Intersection
    else if question is yes/no question **then**
        Determine if question asks for logical comparison or string comparison
        if question asks for logical comparison **then**
            if *either* question **then** discrete_operation $\leftarrow$ Or
            else if *both* question **then** discrete_operation $\leftarrow$ And
        else if question asks for string comparison **then**
            if asks for same? **then** discrete_operation $\leftarrow$ Is equal
            else if asks for difference? **then** discrete_operation $\leftarrow$ Not equal
    return discrete_operation

5abce73055429959677d6b34,5a80071f5542992bc0c4a684,5a840a9e5542992ef85e2397,5a7e02cf5542997cc2c474f4,5ac1c9a15542994ab5c67e1c
5a81ea115542995ce29dcc78,5ae7308d5542991e8301cbb8,5ae527945542993aec5ec167,5ae748d1554299572ea547b0,5a71148b5542994082a3e567
5ae531695542990ba0bbb1fb,5a8f5273554299458435d5b1,5ac2db67554299657fa290a6,5ae0c7e755429945ae95944c,5a7150c75542994082a3e7be
5abffc0d5542990832d3a1e2,5a721bbc55429971e9dc9279,5ab57fc4554299488d4d99c0,5abbda84554299642a094b5b,5ae7936d5542997ec27276a7
5ab2d3df554299194fa9352c,5ac279345542990b17b153b0,5ab8179f5542990e739ec817,5ae20cd25542997283cd2376,5ae67def5542991bbc9760f3
5a901b985542995651fb50b0,5a808cbd5542996402f6a54b,5a8457445542993344760e6,5ab9b1fd5542996be202058e,5a7f1ad155429934daa2fce2
5ade03da5542997dc7907120,5a809fe75542996402f6a5ba,5ae28058554299495565da90,5abd09585542996e802b469b,5a7f9cbd5542994857a7677c
5a7b4073554299042af8f733,5ac119335542992a796dede4,5a7e1a2955429965cec5ea5d,5a8febb555429916514e73e4,5a87184a5542991e771816c5
5a86681c5542991e77181644,5abba584554299642a094afa,5add39e75542997545bbbcc4,5a7f354b5542992e7d278c8c,5a89810655429946c8d6e929
5a78c7db55429974737f7882,5a8d0c1b5542994ba4e3dbb3,5a87e5345542993e715abffb,5ae736cb5542991bbc9761c2,5ae057fd55429945ae959328

Table 11: Question IDs from a set of samples used for ablations in Section 4.6.