A Data Collection Details

A.1 Data Preprocessing

We downloaded the dump of English Wikipedia of October 1, 2017, and extracted text and hyperlinks with WikiExtractor.[8] We use Stanford CoreNLP 3.8.0 (Manning et al., 2014) for word and sentence tokenization. We use the resulting sentence boundaries for collection of supporting facts, and use token boundaries to check whether Turkers are providing answers that cover spans of entire tokens to avoid nonsensical partial-word answers.

A.2 Further Data Collection Details

Details on Curating Wikipedia Pages. To make sure the sampled candidate paragraph pairs are intuitive for crowd workers to ask high-quality multi-hop questions about, we manually curate 591 categories from the lists of popular pages by WikiProject.[9] For each category, we sample (a, b) pairs from the graph G where b is in the considered category, and manually check whether a multi-hop question can be asked given the pair (a, b). Those categories with a high probability of permitting multi-hop questions are selected.

Bonus Structures. To incentivize crowd workers to produce higher-quality data more efficiently, we follow Yang et al. (2018), and employ bonus structures. We mix two settings in our data collection process. In the first setting, we reward the top (in terms of numbers of examples) workers every 200 examples. In the second setting, the workers get bonuses based on their productivity (measured as the number of examples per hour).

A.3 Crowd Worker Interface

Our crowd worker interface is based on ParlAI (Miller et al., 2017), an open-source project that facilitates the development of dialog systems and data collection with a dialog interface. We adapt ParlAI for collecting question answer pairs by converting the collection workflow into a system-oriented dialog. This allows us to have more control over the turkers input, as well as provide turkers with in-the-loop feedbacks or helpful hints to help Turkers finish the task, and therefore speed up the collection process.

Please see Figure 4 for an example of the worker interface during data collection.

[8] https://github.com/attardi/wikiextractor

[9] https://wiki.sh/y8qu

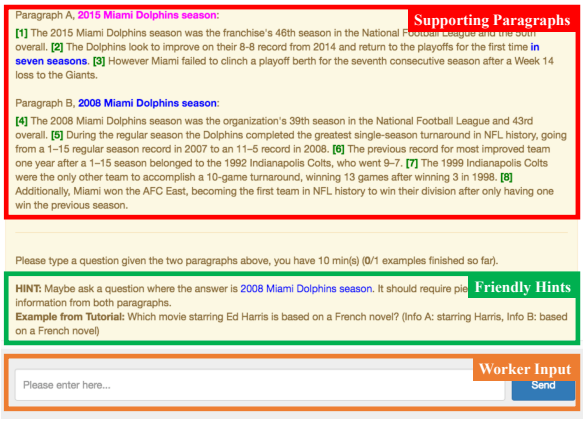

Figure 4: Screenshot of our worker interface on Amazon Mechanical Turk.

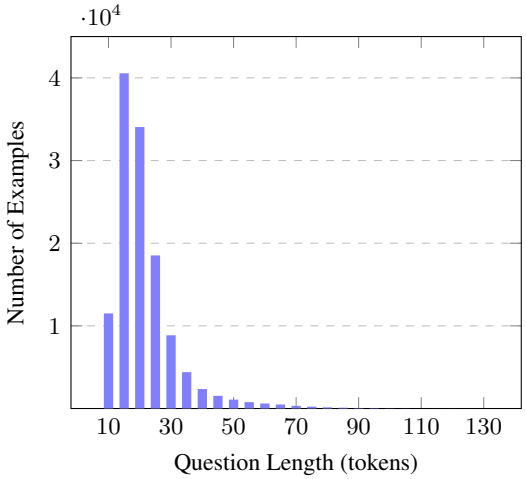

Figure 5: Distribution of lengths of questions in HotpotQA.

B Further Data Analysis

To further look into the diversity of the data in HotpotQA, we further visualized the distribution of question lengths in the dataset in Figure 5. Besides being diverse in terms of types as is show in the main text, questions also vary greatly in length, indicating different levels of complexity and details covered.

C Full Wiki Setting Details

C.1 The Inverted Index Filtering Strategy

In the full wiki setting, we adopt an efficient inverted-index-based filtering strategy for preliminary candidate paragraph retrieval. We provide details in Algorithm 2, where we set the control threshold $N = 5000$ in our experiments. For some of the question q, its corresponding gold para-

Algorithm 2 Inverted Index Filtering Strategy

Input: question text q, control threshold N, ngram-to-Wikidoc inverted index $\mathcal{D}$
Inintialize:
Extract unigram + bigram set r_q from q
$N_{cand} = +\infty$
$C_{gram} = 0$
while $N_{cands} > N$ **do**
  $C_{gram} = C_{gram} + 1$
  Set $S_{overlap}$ to be an empty dictionary
  for $w \in r_q$ **do**
    for $d \in \mathcal{D}[w]$ **do**
      if d not in $S_{overlap}$ **then**
        $S_{overlap}[d] = 1$
      else
        $S_{overlap}[d] = S_{overlap}[d] + 1$
      end if
    end for
  end for
  $S_{cand} = \emptyset$
  for d in $S_{overlap}$ **do**
    if $S_{overlap}[d] \geq C_{gram}$ **then**
      $S_{cand} = S_{cand} \cup \{d\}$
    end if
  end for
  $N_{cands} = |S_{cand}|$
end while
return S_{cand}

graphs may not be included in the output candidate pool S_{cand}, we set such missing gold paragraph's rank as $|S_{cand}| + 1$ during the evaluation, so MAP and Mean Rank reported in this paper are upper bounds of their true values.

C.2 Compare *train-medium* Split to Hard Ones

Table 9 shows the comparison between *train-medium* split and hard examples like *dev* and *test* under retrieval metrics in full wiki setting. As we can see, the performance gap between *train-medium* split and its *dev*/*test* is close, which implies that *train-medium* split has a similar level of difficulty as hard examples under the full wiki setting in which a retrieval model is necessary as the first processing step.

Set	MAP	Mean Rank	CorAns Rank
train-medium	41.89	288.19	82.76
dev	42.79	304.30	97.93
test	45.92	286.20	74.85

Table 9: Retrieval performance comparison on full wiki setting for *train-medium*, *dev* and *test* with 1,000 random samples each. MAP and are in %. Mean Rank averages over retrieval ranks of two gold paragraphs. CorAns Rank refers to the rank of the gold paragraph containing the answer.