Answering Open-Domain Questions of Varying Reasoning Steps from Text

Peng Qi*♠♡   **Haejun Lee*♣**   **Oghenetegiri "TG" Sido*♠**   **Christopher D. Manning♠**

♠ Computer Science Department, Stanford University

♡ JD AI Research

♣ Samsung Research

{pengqi, osido, manning}@cs.stanford.edu, haejun82.lee@samsung.com

Abstract

We develop a unified system to answer directly from text open-domain questions that may require a varying number of retrieval steps. We employ a single multi-task transformer model to perform all the necessary subtasks—retrieving supporting facts, reranking them, and predicting the answer from all retrieved documents—in an iterative fashion. We avoid crucial assumptions of previous work that do not transfer well to real-world settings, including exploiting knowledge of the fixed number of retrieval steps required to answer each question or using structured metadata like knowledge bases or web links that have limited availability. Instead, we design a system that can answer open-domain questions on any text collection without prior knowledge of reasoning complexity. To emulate this setting, we construct a new benchmark, called BEERQA, by combining existing one- and two-step datasets with a new collection of 530 questions that require three Wikipedia pages to answer, unifying Wikipedia corpora versions in the process. We show that our model demonstrates competitive performance on both existing benchmarks and this new benchmark. We make the new benchmark available at https://beerqa.github.io/.

1 Introduction

Using knowledge to solve problems is a hallmark of intelligence. Since human knowledge is often containned in large text collections, open-domain question answering (QA) is an important means for intelligent systems to make use of the knowledge in large text collections. With the help of large-scale datasets based on Wikipedia (Rajpurkar et al., 2016, 2018) and other large corpora (Trischler et al., 2016; Dunn et al., 2017; Talmor and Berant, 2018), the research community has made substantial progress on tackling this problem in recent years, including

in the direction of complex reasoning over multiple pieces of evidence, or *multi-hop* reasoning (Yang et al., 2018; Welbl et al., 2018; Chen et al., 2020).

Despite this success, most previous systems are developed with, and evaluated on, datasets that contain exclusively *single-hop* questions (ones that require a single document or paragraph to answer) or *two-hop* ones. As a result, their design is often tailored exclusively to single-hop (*e.g.*, Chen et al., 2017; Wang et al., 2018b) or multi-hop questions (*e.g.*, Nie et al., 2019; Min et al., 2019; Feldman and El-Yaniv, 2019; Zhao et al., 2020a; Xiong et al., 2021). Even when the model is designed to work with both, it is often trained and evaluated on exclusively single-hop or multi-hop settings (*e.g.*, Asai et al., 2020). In practice, not only can we not expect open-domain QA systems to receive exclusively single- or multi-hop questions from users, but it is also non-trivial to judge reliably whether a question requires one or multiple pieces of evidence to answer *a priori*. For instance, *"In which U.S. state was Facebook founded?"* appears to be single-hop, but its answer cannot be found in the main text of a single English Wikipedia page.

Besides the impractical assumption about reasoning hops, previous work often also assumes access to non-textual metadata such as knowledge bases, entity linking, and Wikipedia hyperlinks when retrieving supporting facts, especially in answering complex questions (Nie et al., 2019; Feldman and El-Yaniv, 2019; Zhao et al., 2019; Asai et al., 2020; Dhingra et al., 2020; Zhao et al., 2020a). While this information is helpful, it is not always available in text collections we might be interested in getting answers from, such as news or academic research articles, besides being labor-intensive and time-consuming to collect and maintain. It is therefore desirable to design a system that is capable of extracting knowledge from text without using such metadata, to maximally emphasize using knowledge available to us in the form of text.

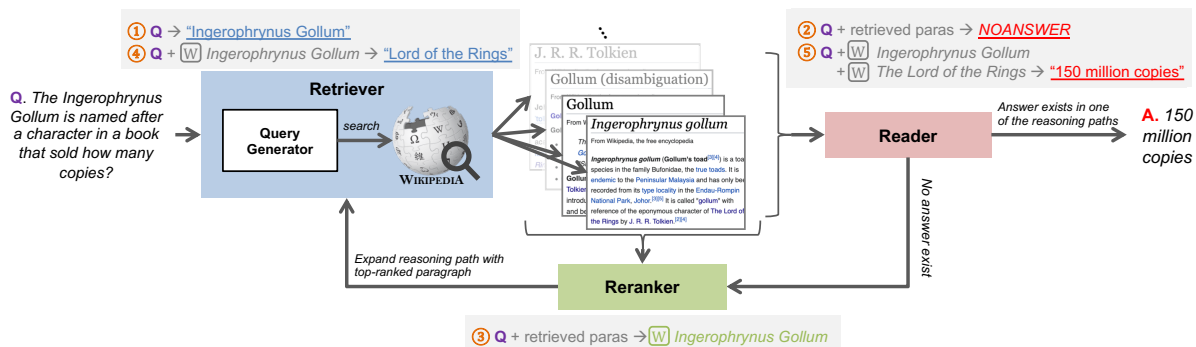
*These authors contributed equally.

Figure 1: The IRRR question answering pipeline answers a complex question in the HotpotQA dataset by iteratively retrieving, reading, and reranking paragraphs from Wikipedia. In this example, the question is answered in five steps: 1. the retriever model selects the words "Ingerophrynus gollum" from the question as an initial search query; 2. the question answering model attempts to answer the question by combining the question with each of the retrieved paragraphs and fails to find an answer; 3. the reranker picks the paragraph about the *Ingerophrynus gollum* toad to extend the reasoning path; 4. the retriever generates an updated query "Lord of the Rings" to retrieve new paragraphs; 5. the reader correctly predicts the answer "150 million copies" by combining the reasoning path (question + "Ingerophrynus gollum") with the newly retrieved paragraph about "The Lord of the Rings".

To address these limitations, we propose Iterative Retriever, Reader, and Reranker (IRRR), which features a single neural network model that performs all of the subtasks required to answer questions from a large collection of text (see Figure 1). IRRR is designed to leverage off-the-shelf information retrieval systems by generating natural language search queries, which allows it to easily adapt to arbitrary collections of text without requiring well-tuned neural retrieval systems or extra metadata. This further allows users to understand and control IRRR, if necessary, to facilitate trust. Moreover, IRRR iteratively retrieves more context to answer the question, which allows it to easily accommodate questions of different number of reasoning steps.

To evaluate the performance of open-domain QA systems in a more realistic setting, we construct a new benchmark called BEERQA[1] by combining the questions from the single-hop SQuAD Open (Rajpurkar et al., 2016; Chen et al., 2017) and the two-hop HotpotQA (Yang et al., 2018) with a new collection of 530 human-annotated questions that require information from at least three Wikipedia pages to answer. We map all questions to a unified version of the English Wikipedia to reduce stylistic differences that might provide statistical shortcuts to models. As a result, BEERQA provides a more realistic evaluation of open-ended question answering systems in their ability to answer questions without knowledge of the number of reasoning steps required ahead of time. We show that IRRR not

only achieves competitive performance with state-of-the-art models on the original SQuAD Open and HotpotQA datasets, but also establishes a strong baseline for this new dataset.

To recap, our contributions in this paper are: (1) a new open-domain QA benchmark, BEERQA, that features questions requiring variable steps of reasoning to answer on a unified Wikipedia corpus. (2) A single unified neural network model that performs all essential subtasks in open-domain QA purely from text (retrieval, reranking, and reading comprehension), which not only achieves strong results on SQuAD and HotpotQA, but also establishes a strong baseline on this new benchmark.[2]

2 Open-Domain Question Answering

The task of open-domain question answering is concerned with finding the answer a to a question q from a large text collection $\mathcal{D}$. Successful solutions to this task usually involve two crucial components: an *information retrieval* system that finds a small set of relevant documents $\mathcal{D}_r$ from $\mathcal{D}$, and a *reading comprehension* system that extracts the answer from it.[3] Chen et al. (2017) presented the first neural-network-based approach to this problem, which was later extended by Wang et al. (2018a) with a *reranking* system to further reduce the amount of

---

[1] https://beerqa.github.io/

[2] Our code for the model can be found at: https://github.com/beerqa/IRRR.

[3] Some recent work breaks away from this mold, and use large pretrained language models (*e.g.*, T5; Raffel et al., 2020) to directly generate answers from knowledge stored in model parameters.

context the reading comprehension component has to consider to improve answer accuracy.

More recently, Yang et al. (2018) showed that this single-step retrieve-and-read approach to open-domain question answering is inadequate for more complex questions that require multiple pieces of evidence to answer (*e.g.*, *"What is the population of Mark Twain's hometown?"*). While later work approaches these by extending supporting fact retrieval beyond one step, most assumes that all questions are either exclusively single-hop or multi-hop during training and evaluation. We propose IRRR, a system that performs variable-hop retrieval for open-domain QA to address these issues, and present a new benchmark, BEERQA, to evaluate systems in a more realistic setting.

3 IRRR: Iterative Retriever, Reader, and Reranker

In this section, we present a unified model to perform all of the subtasks necessary for open-domain question answering—Iterative Retriever, Reader, and Reranker (IRRR), which performs the subtasks involved in an iterative manner to accommodate questions with a varying number of steps. IRRR aims at building a reasoning path p from the question q, through all the necessary supporting documents or paragraphs $d \in \mathcal{D}_{\text{gold}}$ to the answer a (where $\mathcal{D}_{\text{gold}}$ is the set of gold supporting facts).[4] As shown in Figure 1, IRRR operates in a loop of retrieval, reading, and reranking to expand the reasoning path p with new documents from $d \in \mathcal{D}$.

Specifically, given a question q, we initialize the reasoning path with the question itself, *i.e.*, $p_0 = [q]$, and generate from it a search query with IRRR's retriever. Once a set of relevant documents $\mathcal{D}_1 \subset \mathcal{D}$ is retrieved, they might either help answer the question, or reveal clues about the next piece of evidence we need to answer q. The reader model then attempts to read each of the documents in $\mathcal{D}_1$ to answer the question combined with the current reasoning path p. If more than one answer can be found from these candidate reasoning paths, we predict the answer with the highest *answerability* score, which we will detail in section 3.2. If no answer can be found, then IRRR's reranker scores each retrieved paragraph against the current reasoning path, and appends the top-ranked paragraph to the current reasoning path,

---

[4]For simplicity, we assume that there is a single set of relevant supporting facts that helps answer each question.
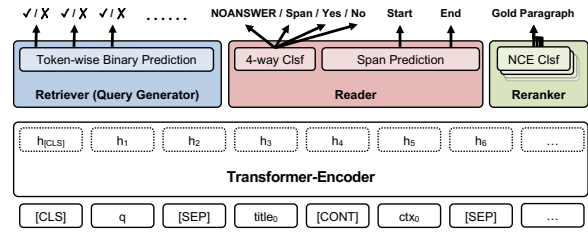

Figure 2: The overall architecture of our IRRR model, which uses a shared Transformer encoder to perform all subtasks of open-domain question answering.

i.e., $p_{i+1} = p_i + [\arg\max_{d \in D_1} \text{reranker}(p_i, d)]$, before the updated reasoning path is presented to the retriever to generate new search queries. This iterative process is repeated until an answer is predicted from one of the reasoning paths, or until the reasoning path has reached a cap of K documents.

To reduce computational cost and improve model representations of reasoning paths from shared statistical learning, IRRR is implemented as a multi-task model built on a pretrained Transformer model that performs all three subtasks. At a high level, it consists of a Transformer encoder (Vaswani et al., 2017) which takes the reasoning path p (the question and all retrieved paragraphs so far) as input, and one set of task-specific parameters for each task of retrieval, reranking, and reading comprehension (see Figure 2). The retriever generates natural language search queries by selecting words from the reasoning path, the reader extracts answers from the reasoning path and abstains if its confidence is not high enough, and the reranker assigns a scalar score for each retrieved paragraph as a potential continuation of the current reasoning path.

The input to our Transformer encoder is formatted similarly to that of the BERT model (Devlin et al., 2019). For a reasoning path p that consists of the question and t retrieved paragraphs, the input is formatted as "[CLS] question [SEP] title_1 [CONT] para_1 [SEP] ... title_t [CONT] para_t [SEP]", where [CLS], [SEP], and [CONT] are special tokens to separate different components of the input. The [CONT] embedding is randomly initialized with a truncated normal distribution with a standard deviation of 0.02, and finetuned with other model parameters during training.

We will detail each of the task-specific components in the following subsections.

3.1 Retriever

The goal of the retriever is to generate natural language queries to retrieve relevant documents from an off-the-shelf text-based retrieval engine.[5] This allows IRRR to perform open-domain QA in an explainable and controllable manner, where a user can easily understand the model's behavior and intervene if necessary. We extract search queries from the current reasoning path, *i.e.*, the original question and all of the paragraphs that we have already retrieved, similar to GOLDEN Retriever's approach (Qi et al., 2019). This is based on the observation that there is usually a strong semantic overlap between the reasoning path and the next paragraph to retrieve, which helps reduce the search space of potential queries. We note, though, that IRRR differs from GOLDEN Retriever in two important ways: (1) we allow search queries to be any subsequence of the reasoning path instead of limiting it to substrings to allow for more flexible combinations of search phrases; (2) more importantly, we employ the same retriever model across reasoning steps to generate queries instead of training separate ones for each reasoning step, which is crucial for IRRR to generalize to arbitrary reasoning steps.

To predict these search queries from the reasoning path, we apply a token-wise binary classifier on top of the shared Transformer encoder model, to decide whether each token is included in the final query. At training time, we derive supervision signal to train these classifiers with a binary cross entropy loss (which we detail in Section 3.4.1); at test time, we select a cutoff threshold for query words to be included from the reasoning path. In practice, we find that boosting the model to predict more query terms is beneficial to increase the recall of the target paragraphs in retrieval.

3.2 Reader

The reader model attempts to find the answer given a reasoning path comprised of the question and retrieved paragraphs. To support unanswerable questions and the special non-extractive answers *yes* and *no* from HotpotQA, we train a classifier conditioned on the Transformer encoder representation of the `[CLS]` token to predict one of the 4 classes `SPAN/YES/NO/NOANSWER`. The classifier thus simultaneously assigns an *answerability* score

to this reasoning path to assess the likelihood of the document having the answer to the original question on this reasoning path. Span answers are predicted from the context using a span start classifier and a span end classifier, following Devlin et al. (2019).

We define answerability as the log likelihood ratio between the most likely positive answer and the `NOANSWER` prediction, and use it to pick the best answer from all the candidate reasoning paths to stop IRRR's iterative process, if found. We find that this likelihood ratio formulation is less affected by sequence length compared to prediction probability, thus making it easier to assign a global threshold across reasoning paths of different lengths to stop further retrieval. We include further details about answerability calculation in Appendix C.

3.3 Reranker

When the reader fails to find an answer from the reasoning path, the reranker selects one of the retrieved paragraphs to expand it, so that the retriever can generate new search queries to retrieve new context to answer the question. To achieve this, we assign each potential extended reasoning path a score by linearly transforming the hidden representation of the `[CLS]` token, and picking the extension that has the highest score. At training time, we normalize the reranker scores across top retrieved paragraphs with softmax, and maximize the log likelihood of selecting gold supporting paragraphs from retrieved ones, which is a noise contrastive estimation (NCE; Mnih and Kavukcuoglu, 2013; Jean et al., 2015) of the reranker likelihood over all retrieved paragraphs.

3.4 Training IRRR

3.4.1 Dynamic Oracle for Query Generation

Since existing open-domain QA datasets do not include human-annotated search queries, we need to derive supervision signal to train the retriever with a dynamic oracle. Similar to GOLDEN Retriever, we derive search queries from overlapping terms between the reasoning path and the target paragraph with the goal of maximizing retrieval performance.

To reduce computational cost, we limit our attention to overlapping spans of text between the reasoning path and the target document when generating oracle queries. For instance, when "David" is part of the overlapping span "David Dunn", the entire span is either included or excluded from the oracle query to reduce the search space. Once N

---

[5]We employ Elasticsearch (Gormley and Tong, 2015) as our text-based search engine, and follow previous work to process Wikipedia and search results, which we detail in Appendix B.
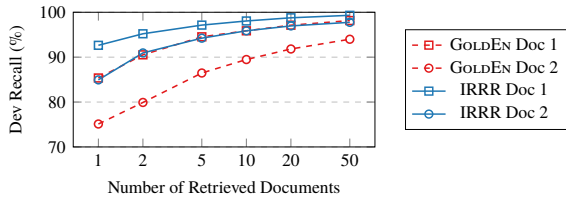
Figure 3: Recall of the two gold supporting documents by the oracle queries of GOLDEN Retriever and IRRR on the HotpotQA dataset, where each question corresponds to two supporting documents.

Question: How many counties are on the island that is home to the fictional setting of the novel in which Daisy Buchanan is a supporting character?
Wikipedia Page 1: *Daisy Buchanan*
Daisy Fay Buchanan is a fictional character in F. Scott Fitzgerald's magnum opus "The Great Gatsby" (1925)...
Wikipedia Page 2: *The Great Gatsby*
The Great Gatsby is a 1925 novel ... that follows a cast of characters living in the fictional town of West Egg on prosperous Long Island ...
Wikipedia Page 3: *Long Island*
The Long Island ... comprises four counties in the U.S. state of New York: Kings and Queens ... to the west; and Nassau and Suffolk to the east...
Answer: four

Figure 4: An example of the newly collected challenge questions. This particular question requires three pieces of evidence to answer.

overlapping spans are found, we approximate the importance of each with the following "importance" metric to avoid enumerating all 2^N combinations to generate the oracle query

$$\text{Imp}(s_i) = \text{Rank}(t, \{s_j\}_{j=1, j \neq i}^N) - \text{Rank}(t, \{s_i\}),$$

where s_j are overlapping spans, and $\text{Rank}(t, S)$ is the rank of target document t in the search result when spans S are used as search queries (the smaller, the closer t is to the top). Intuitively, the second term captures the importance of the search term when used alone, and the first captures its importance when combined with all other overlapping spans, which helps us capture query terms that are only effective when combined. After estimating importance of each overlapping span, we determine the final oracle query by first sorting all spans by descending importance, then including each in the final oracle query until the search rank of t stops improving. The resulting time complexity for generating these oracle queries is thus $O(N)$, *i.e.*, linear in the number of overlapping spans between the reasoning path and the target paragraph.

Figure 3 shows that the added flexibility of non-span queries in IRRR significantly improves retrieval performance compared to that of GOLDEN Retriever, which is only able to extract contiguous spans from the reasoning path as queries.

3.4.2 Reducing Exposure Bias with Data Augmentation

With the dynamic oracle, we are able to generate target queries to train the retriever model, retrieve documents to train the reranker model, and expand reasoning paths in the training set by always choosing a gold paragraph, following Qi et al. (2019). However, this might prevent the model from generalizing to cases where model behavior deviates from the oracle. To address this, we augment the training data by occasionally selecting non-gold

paragraphs to expand reasoning paths, and use the dynamic oracle to generate queries for the model to "recover" from these synthesized retrieval mistakes. We found that this data augmentation significantly improves the performance of IRRR in preliminary experiments, and thus report main results with augmented training data.

4 Experiments

Standard Benchmarks. We test IRRR on two standard benchmarks, SQuAD Open and HotpotQA. SQuAD Open (Chen et al., 2017) designates the development set of the original SQuAD dataset as its test set, which features more than 10,000 questions, each based on a single paragraph in a Wikipedia article. For this dataset, we follow previous work and use the 2016 English Wikipedia as the corpus for evaluation. Since the authors did not present a standard development set, we further split part of the training set to construct a development set roughly as large as the test set. HotpotQA (Yang et al., 2018) features more than 100,000 questions that require the introductory paragraphs of two Wikipedia articles to answer, and we focus on its open-domain "fullwiki" setting in this work. For HotpotQA, we use the introductory paragraphs provided by the authors for training and evaluation, which is based on a 2017 Wikipedia dump.

New Benchmark. To evaluate the performance of IRRR as well as future QA systems in a more realistic open-domain setting without a pre-specified number of reasoning steps for each question, we further combine SQuAD Open and HotpotQA with 530 newly collected challenge questions (see Figure 4 for an example, and Appendix E for more details)

	SQuAD Open	HotpotQA	3+ Hop	Total
Train	59,285	74,758	0	134,043
Dev	8,132	5,989	0	14,121
Test	8,424	5,978	530	14,932
Total	75,841	86,725	530	163,096

Table 1: Counts of QA examples in the new unified benchmark, BEERQA.

to construct a new benchmark. Note that naively combining the datasets by merging the questions and the underlying corpora is problematic, as the corpora not only feature repeated and sometimes contradicting information, but also make them available in two distinct forms (full Wikipedia pages in one and just the introductory paragraphs in the other). This could result in models taking corpus style as a shortcut to determine question complexity, or even result in plausible false answers due to corpus inconsistency.

To construct a high-quality unified benchmark, we begin by mapping the paragraphs each question is based on to a more recent version of Wikipedia.[6] We discarded examples where the Wikipedia pages have either been removed or significantly edited such that the answer can no longer be found from paragraphs that are similar enough to the original contexts the questions are based on.[7] As a result, we filtered out 22,328 examples from SQuAD Open, and 18,649 examples from HotpotQA's fullwiki setting. We add newly annotated challenge questions to the test set of the new benchmark, which require at least three steps of reasoning to answer. This allows us to test the generalization capabilities of QA models to this unseen scenario. The statistics of the final dataset, which we name BEERQA, can be found in Table 1. For all benchmark datasets, we report standard answer exact match (EM) and unigram F_1 metrics.

Training details. We use ELECTRA$_{\text{LARGE}}$ (Clark et al., 2020) as the pre-trained initialization for our Transformer encoder. We train the model on a combined dataset of SQuAD Open and HotpotQA questions where we optimize the joint loss of the retriever, reader, and reranker components simultaneously in an multi-task learning

fashion. Training data for the retriever and reranker components is derived from the dynamic oracle on the training set of these datasets, where reasoning paths are expanded with oracle queries and by picking the gold paragraphs as they are retrieved for the reader component. We augment the training data with the technique in Section 3.4.2 and expand reasoning paths up to 3 reasoning steps on HotpotQA and 2 on SQuAD Open, and find that this results in a more robust model. After an initial model is finetuned on this expanded training set, we apply our iterative training technique to further reduce exposure bias of the model by generating more data with the trained model and the dynamic oracle.

5 Results

In this section, we present the performance of IRRR when evaluated against previous systems on standard benchmarks, and demonstrate its efficacy on our new, unified benchmark, especially with the help of iterative training.

5.1 Performance on Standard Benchmarks

We first compare IRRR against previous systems on SQuAD Open and the fullwiki setting of HotpotQA. On each dataset, we compare the performance of IRRR against best previously published systems, as well as unpublished ones on public leaderboards. For a fairer comparison to previous work, we make use of their respective Wikipedia corpora, and limit the retriever to retrieve 150 paragraphs of text from Wikipedia at each step of reasoning. We also compare IRRR against the Graph Recurrent Retriever (GRR; Asai et al., 2020) on our newly collected 3-hop question challenge test set, using the author's released code and models trained on HotpotQA. In these experiments, we report IRRR performance both from training on the dataset it is evaluated on, and from combining the training data we derived from both SQuAD Open and HotpotQA.

As can be seen in Tables 2 and 3, IRRR achieves competitive performance with previous work, and further outperforms previously published work on SQuAD Open by a large margin when trained on combined data. It also outperforms systems that were submitted after IRRR was initially submitted to the HotpotQA leaderboard. On the 3+ hop challenge set, we similarly notice a large performance margin between IRRR and GRR, although neither is trained with questions requiring three or more hops, demonstrating that IRRR generalizes well to

System	SQuAD Open	
	EM	F_1
DrQA (Chen et al., 2017)	27.1	—
DensePR (Karpukhin et al., 2020)	38.1	—
BERTserini (Yang et al., 2019)	38.6	46.1
MUPPET (Feldman and El-Yaniv, 2019)	39.3	46.2
RE^3 (Hu et al., 2019)	41.9	50.2
Knowledge-aided (Zhou et al., 2020)	43.6	53.4
Multi-passage BERT (Wang et al., 2019)	53.0	60.9
GRR (Asai et al., 2020)	56.5	63.8
FiD (Izacard and Grave, 2020)	56.7	—
SPARTA (Zhao et al., 2020b)	59.3	66.5
IRRR (SQuAD)	56.8	63.2
IRRR (SQuAD+HotpotQA)	**61.8**	**68.9**

Table 2: End-to-end question answering performance on SQuAD Open, evaluated on the same set of documents as Chen et al. (2017).

System	HotpotQA		3+ hop	
	EM	F_1	EM	F_1
GRR (Asai et al., 2020)	60.0	73.0	27.2[†]	31.9[†]
Step-by-step[⊗]	63.0	75.4	—	—
DDRQA (Zhang et al., 2021)	62.3	75.3	—	—
MDR (Xiong et al., 2021)	62.3	75.3	—	—
EBS-SH [⊗]	65.5	78.6	—	—
TPRR [⊗]	67.0	79.5	—	—
HopRetriever (Li et al., 2020)	67.1	79.9	—	—
IRRR (HotpotQA)	65.2	78.0	29.2	34.2
IRRR (SQuAD + HotpotQA)	65.7	78.2	**32.5**	**36.7**

Table 3: End-to-end question answering performance on HotpotQA and the new 3+ hop challenge questions, evaluated on the official HotpotQA Wikipedia paragraphs. ⊗ denotes anonymous/preprint unavailable at the time of writing of this paper. † indicates results we obtained using the publicly available code and pre-trained models.

questions that require more retrieval steps than the ones seen during training. We note that the systems that outperform IRRR on these datasets typically make use of trainable neural retrieval components, which IRRR can potentially benefit from adopting as well. Specifically, SPARTA (Zhao et al., 2020b) introduces a neural sparse retrieval system that potentially works well with IRRR's oracle query generation procedure to further improve retrieval performance, thanks to its use of natural language queries. HopRetriever (Li et al., 2020) introduces a novel representation of documents for retrieval that is particularly suitable for discovering documents connected by the same entity to answer multi-hop questions, which IRRR could benefit from as well. We leave exploration of these directions to future work.
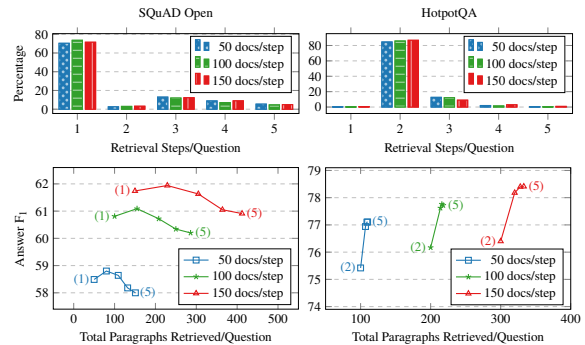
To better understand the behavior of IRRR on

Figure 5: The retrieval behavior of IRRR and its relation to the performance of end-to-end question answering. Top: The distribution of reasoning path lengths as determined by IRRR. Bottom: Total number of paragraphs retrieved by IRRR vs. the end-to-end question answering performance as measured by answer F_1.

these benchmarks, we analyze the number of paragraphs retrieved by the model when varying the number of paragraphs retrieved at each reasoning step among $\{50, 100, 150\}$. As can be seen in Figure 5, IRRR stops its iterative process as soon as all necessary paragraphs to answer the question have been retrieved, effectively reducing the total number of paragraphs retrieved and read by the model compared to always retrieving a fixed number of paragraphs for each question. Further, we note that the optimal cap for the number of reasoning steps is larger than the number of gold paragraphs necessary to answer the question on each benchmark, which we find is due to IRRR's ability to recover from retrieving and selecting non-gold paragraphs (see the example in Figure 6). Finally, we note that increasing the number of paragraphs retrieved at each reasoning step remains an effective, if computationally expensive, strategy to improve the end-to-end performance of IRRR. However, the tradeoff between retrieval budget and model performance is more effective than that of previous work (*e.g.*, GRR), and we note that the queries generated by IRRR are explainable to humans and can help humans easily control its behavior.

5.2 Performance on the Unified Benchmark

To demonstrate the performance of IRRR in a more realistic setting of open-domain QA, we evaluate it on the new, unified benchmark. As is shown in Table 4, IRRR's performance remains competitive on all questions from different origins in the unified benchmark, despite the difference in reasoning complexity when answering these questions.

	Dev		Test	
	EM	F_1	EM	F_1
SQuAD Open	50.65	60.99	60.59	67.51
HotpotQA	59.01	70.33	58.61	69.86
3+ hop	—	—	33.02	39.59
Micro-averaged	54.20	64.95	58.82	67.46
Macro-averaged	54.83	65.66	50.74	58.99

Table 4: End-to-end question answering performance of IRRR on the unified benchmark, evaluated on the 2020 copy of Wikipedia. These results are not directly comparable with those in Tables 2 and 3 because the set of questions and Wikipedia documents differ.

System	SQuAD	HotpotQA
Ours (joint dataset)	58.69	68.74
vs. fixed retrieval steps ($K = 3$)	31.70	66.60
vs. remove HotpotQA / SQuAD data	54.35	66.91
replace ELECTRA w/ BERT$_{\text{LARGE-WWM}}$	57.19	63.86

Table 5: Ablation study of different design choices in IRRR, as evaluated by Answer F_1 on the dev set of the unified benchmark. Results differ from those in Table 4 because fewer reasoning steps are used (3 vs. 5) and fewer paragraphs retrieved at each step (50 vs. 150).

The model also generalizes to the 3-hop questions despite having never been trained on them. We note that the large performance gap between the development and test settings for SQuAD Open questions is due to the fact that test set questions (the original SQuAD dev set) are annotated with multiple human answers, while the dev set ones (originally from the SQuAD training set) are not.

To better understand the contribution of the various components and techniques we proposed for IRRR, we performed ablation studies on the model iterating up to 3 reasoning steps with 50 paragraphs for each step, and present the results in Table 5. First of all, we find it is important to allow IRRR to dynamically stop retrieving paragraphs to answer the question. Compared to its fixed-step retrieval counterpart, dynamically stopping IRRR improves F_1 on both SQuAD and HotpotQA questions by 27.0 and 2.1 points respectively (we include further analyses for dynamic stopping in Appendix D). We also find combining SQuAD and HotpotQA datasets beneficial for both datasets in an open-domain setting, and that ELECTRA is an effective alternative to BERT for this task.

6 Related Work

The availability of large-scale question answering (QA) datasets has greatly contributed to the research progress on open-domain QA. SQuAD (Rajpurkar

Question	The **Ingerophrynus gollum** is _named_ after a **character** in a **book** that _sold_ how many **copies**?
Step 1 (Non-Gold)	Ingerophrynus is a genus of true toads with 12 species. ... In 2007 a new species, "Ingerophrynus gollum", was added to this genus. This species is named after the character Gollum created by **J. R. R. Tolkien.**"
Query	Ingerophrynus gollum book sold copies **J. R. R. Tolkien**
Step 2 (Gold)	Ingerophrynus gollum (Gollum's toad) is a species of _true_ toad. ... It is called "gollum" with reference of the eponymous character of The **Lord of the Rings** by J. R. R. Tolkien.
Query	Ingerophrynus gollum character book sold copies J. R. R. Tolkien **true Lord of the Rings**
Step 3 (Gold)	The Lord of the Rings is an epic high fantasy novel written by English author and scholar J. R. R. Tolkien. ... is one of the best-selling novels ever written, with **150 million copies** sold.
Answer/GT	**150 million copies**

Figure 6: An example of IRRR answering a question from HotpotQA by generating natural language queries to retrieve paragraphs, then rerank them to compose reasoning paths and read them to predict the answer. Here, IRRR recovers from an initial retrieval/reranking mistake by retrieving more paragraphs, before arriving at the gold supporting facts and the correct answer.

et al., 2016, 2018) is among the first question answering datasets adopted for this purpose by Chen et al. (2017) to build QA systems over Wikipedia articles. Similarly, TriviaQA (Joshi et al., 2017) and Natural Questions (Kwiatkowski et al., 2019) feature Wikipedia-based questions that are written by trivia enthusiasts and extracted from Google search queries, respectively. More recently, Petroni et al. (2021) presented, KILT, a new benchmark based on Wikipedia where many knowledge-based tasks are evaluated in a unified version of Wikipedia, including open-domain question answering, entity linking, dialogue, etc. Unlike BEERQA, however, single-hop and multi-hop QA are held completely separate during evaluation in KILT, which makes the evaluation of open-domain QA less realistic. Aside from Wikipedia, researchers have also used news articles (Trischler et al., 2016) and search results from the web (Dunn et al., 2017; Talmor and Berant, 2018) as the corpus for open-domain QA.

Inspired by the TREC QA challenge,[8] Chen et al. (2017) were the first to combine information retrieval systems with accurate neural network-based reading comprehension models for open-domain QA. Recent work has improved open-domain QA performance by enhancing various components in this retrieve-and-read approach. While much research focused on improving the reading comprehension model (Seo et al., 2017; Clark and Gardner, 2018), especially with pretrained langauge models like BERT (Devlin et al., 2019), researchers have

---

[8] https://trec.nist.gov/data/qamain.html

also demonstrated that neural network-based information retrieval systems achieve competitive, if not better, performance compared to traditional IR engines (Lee et al., 2019; Khattab et al., 2020; Guu et al., 2020; Xiong et al., 2021). Aside from the reading comprehension and retrieval components, researchers have also found value from reranking search results (Wang et al., 2018a) or answer candidates (Wang et al., 2018b; Hu et al., 2019).

While most work focuses on questions that require only a local context of supporting facts to answer, Yang et al. (2018) presented HotpotQA, which tests whether open-domain QA systems can generalize to more complex questions that require evidence from multiple documents to answer. Researchers have explored various techniques to extend retrieve-and-read systems to this problem, including making use of hyperlinks between Wikipedia articles (Nie et al., 2019; Feldman and El-Yaniv, 2019; Zhao et al., 2019; Asai et al., 2020; Dhingra et al., 2020; Zhao et al., 2019) and iterative retrieval (Talmor and Berant, 2018; Das et al., 2019; Qi et al., 2019). While most previous work on iterative retrieval makes use of neural retrieval systems that directly accept real vectors as input, our work is similar to that of Qi et al. (2019) in using natural language search queries. A crucial distinction between our work and previous work on multi-hop open-domain QA, however, is that we don't train models to exclusively answer single-hop or multi-hop questions, but demonstrate that one single set of parameters performs well on both tasks.

7 Conclusion

In this paper, we presented Iterative Retriever, Reader, and Reranker (IRRR), a system that uses a single model to perform subtasks to answer open-domain questions of arbitrary reasoning steps. IRRR achieves competitive results on standard open-domain QA benchmarks, and establishes a strong baseline on BEERQA, the new unified benchmark we present, which features questions with mixed levels of complexity.

Acknowledgments

References

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

Giusepppe Attardi. 2015. WikiExtractor. https://github.com/attardi/wikiextractor.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. 2020. HybridQA: A dataset of multi-hop question answering over tabular and textual data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.

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

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. 2019. Multi-step retriever-reader interaction for scalable open-domain question answering. In *International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

Bhuwan Dhingra, Manzil Zaheer, Vidhisha Balachandran, Graham Neubig, Ruslan Salakhutdinov, and William W. Cohen. 2020. Differentiable reasoning over a virtual knowledge base. In *International Conference on Learning Representations*.

Matthew Dunn, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. SearchQA: A new Q&A dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*.

Yair Feldman and Ran El-Yaniv. 2019. Multi-hop paragraph retrieval for open-domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Clinton Gormley and Zachary Tong. 2015. *Elastic-search: The definitive guide: A distributed real-time search and analytics engine*. O'Reilly Media, Inc.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*.

Minghao Hu, Yuxing Peng, Zhen Huang, and Dongsheng Li. 2019. Retrieve, read, rerank: Towards end-to-end multi-document reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Gautier Izacard and Edouard Grave. 2020. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282*.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

V. Karpukhin, Barlas Ouguz, Sewon Min, Patrick Lewis, Ledell Yu Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv*, abs/2004.04906.

Omar Khattab, Christopher Potts, and Matei Zaharia. 2020. Relevance-guided supervision for OpenQA with ColBERT. *arXiv preprint arXiv:2007.00814*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7.

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

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

Yuanhua Lv and ChengXiang Zhai. 2011. When documents are very long, BM25 fails! In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 1103–1104.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. Multi-hop reading comprehension through question decomposition and rescoring. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6097–6109, Florence, Italy. Association for Computational Linguistics.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in neural information processing systems*, pages 2265–2273.

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

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3. *NIST Special Publication*, pages 109–126.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations*.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 641–651.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. NewsQA: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

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

Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2018b. Evidence aggregation for answer re-ranking in open-domain question answering. In *International Conference on Learning Representations*.

Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. 2019. Multi-passage BERT: A globally normalized BERT model for open-domain question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

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

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

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-end open-domain question answering with BERTserini. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Yuyu Zhang, Ping Nie, Arun Ramamurthy, and Le Song. 2021. IDRQA: Iterative document reranking for open-domain multi-hop question answering. In *SIGIR*.

Chen Zhao, Chenyan Xiong, Xin Qian, and Jordan Boyd-Graber. 2020a. Complex factoid question answering with a free-text knowledge graph. In *Proceedings of The Web Conference 2020*, pages 1205–1216.

Chen Zhao, Chenyan Xiong, Corby Rosset, Xia Song, Paul Bennett, and Saurabh Tiwary. 2019. Transformer-XH: Multi-evidence reasoning with extra hop attention. In *International Conference on Learning Representations*.

Tiancheng Zhao, Xiaopeng Lu, and Kyusong Lee. 2020b. Sparta: Efficient open-domain question answering via sparse transformer matching retrieval. *arXiv preprint arXiv:2009.13013*.

Mantong Zhou, Zhouxing Shi, Minlie Huang, and Xiaoyan Zhu. 2020. Knowledge-aided open-domain question answering. *arXiv preprint arXiv:2006.05244*.

A Data processing

In this section, we describe how we process the English Wikipedia and the SQuAD dataset for training and evaluating IRRR.

For the standard benchmarks (SQuAD Open and HotpotQA fullwiki), we use the Wikipedia corpora prepared by Chen et al. (2017) and Yang et al. (2018), respectively, so that our results are comparable with previous work on these benchmarks. Specifically, for SQuAD Open, we use the processed English Wikipedia released by Chen et al. (2017) which was accessed in 2016, and contains 5,075,182 documents.[9] For HotpotQA, Yang et al. (2018) released a processed set of Wikipedia introductory paragraphs from the English Wikipedia originally accessed in October 2017.[10]

While it is established that the SQuAD dev set is repurposed as the test set for SQuAD Open for ease of evaluation, most previous work make use of the entire training set during training, and as a result a proper development set for SQuAD Open does not exist.[11] We therefore resplit the SQuAD training set into a proper development set that is not used during training, and a reduced training set that we use for all of our experiments. As a result, although IRRR is evaluated on the same test set as previous systems, it is likely disadvantaged due to the reduced amount of training data and hyperparameter tuning on this new dev set. We split the training set by first grouping questions and paragraphs by the Wikipedia entity/title they belong to, then randomly selecting entities to add to the dev set until the dev set contains roughly as many questions as the test set (original SQuAD dev set). The statistics of our resplit of SQuAD can be found in Table 6. We make our resplit publicly available to the community at `https://beerqa.github.io/`.

For the unified benchmark, we started by processing the English Wikipedia[12] with the WikiExtractor (Attardi, 2015). We then tokenized this dump and the supporting context used in SQuAD and HotpotQA with Stanford CoreNLP 4.0.0 (Manning et al., 2014) to look for paragraphs in the

Split	Origin	# Entities	#QAs
train	train	387	77,087
dev	train	55	10,512
test	dev	48	10,570

Table 6: Statistics of the resplit SQuAD dataset for proper training and evaluation on the SQuAD Open setting.

2020 Wikipedia dump that might correspond to the context paragraphs in these datasets. Since many Wikipedia articles have been renamed or removed since, we begin by following Wikipedia redirect links to locate the current title of the corresponding Wikipedia page (*e.g.*, the page *"Madonna (entertainer)"* has been renamed *"Madonna"*). After the correct Wikipedia article is located, we look for combinations of one to two consecutive paragraphs in the 2020 Wikipedia dump that have high overlap with context paragraphs in these datasets. We calculate the recall of words and phrases in the original context paragraph (because Wikipedia paragraphs are often expanded with more details), and pick the best combination of paragraphs from the article. If the best candidate has either more than 66% unigrams in the original context, or if there is a common subsequence between the two that covers more than 50% of the original context, we consider the matching successful, and map the answers to the new context paragraphs. The main causes of mismatches are a) Wikipedia pages that have been permanently removed (due to copyright issues, unable to meet notability standards, etc.); b) significantly edited to improve presentation (see Figure 7(a)); c) significantly edited because the world has changed (see Figure 7(b)).

As a result, 20,182/2,146 SQuAD train/dev examples (that is, 17,802/2,380/2,146 train/dev/test examples after data resplit) and 15,806/1,416/1,427 HotpotQA train/dev/fullwiki test examples have been excluded from the unified benchmark. To understand the data quality after converting SQuAD Open and HotpotQA to the newer version of Wikipedia, we sampled 100 examples from the training split of each dataset. We find that 6% of SQuAD questions and 10% of HotpotQA questions are no longer answerable from their context paragraphs due to edits in Wikipedia or changes in the world, despite the presence of the answer span. We also find that 43% of HotpotQA examples contain more than the minimal set of necessary paragraphs

---

[9]`https://github.com/facebookresearch/DrQA`

[10]`https://hotpotqa.github.io/wiki-readme.html`

[11]Thus, if any hyperparameter tuning has been performed, it is usually done to directly maximize the performance on this held-out test set, inflating the performance on this set as a result.

[12]Accessed on August 1st, 2020, which contains 6,133,150 articles in total.

Madonna Louise Ciccone (born August 16, 1958) is an American singer, songwriter, actress, and <u>business-woman</u>. *She achieved popularity by pushing the boundaries of lyrical content in mainstream popular music and imagery in her music videos*, <u>which became a fixture on MTV</u>. *Madonna is known for reinventing both her music and image, and for maintaining her autonomy within the recording industry.* Music critics have acclaimed her musical productions, which have generated some controversy. Referred to as the "Queen of Pop", Madonna is often cited as an influence by other artists.	Madonna Louise Ciccone (born August 16, 1958) is an American singer-songwriter, author, actress and **record executive**. She has been referred to as the "Queen of Pop" **since the 1980s**. *Madonna is noted for her continual reinvention and versatility in music production, songwriting, and visual presentation. She has pushed the boundaries of artistic expression in popular culture, while remaining completely in charge of every aspect of her career.* Her works, **which incorporate social, political, sexual, and religious themes,** have made a cultural impact which has generated both critical acclaim and controversy. Madonna is often cited as an influence by other artists.

(a) The Wikipedia page about Madonna, on December 20, 2016 (on the left, which is in the version SQuAD Open used) versus July 31, 2020 (on the right, which is in the version BEERQA used).

Peter Langkjær Madsen (born 12 January 1971) is a Danish <u>aerospace engineering enthusiast, "art engineer", submarine builder, entrepreneur, co-founder of the non-profit organization Copenhagen Suborbitals, and founder and CEO of RML Spacelab ApS. He was arrested in August 2017 for involvement</u> in the death of Swedish journalist Kim Wall; <u>the investigation is ongoing.</u>	Peter Langkjær Madsen ([]; born 12 January 1971) is a Danish **convicted murderer. In April 2018 he was convicted of the 2017** murder of Swedish journalist Kim Wall **on board his submarine, UC3 Nautilus, and sentenced to life imprisonment. He had previously been an engineer and entrepreneur.**

(b) The Wikipedia page about Peter Madsen, on September 27, 2017 (on the left, which is in the version HotpotQA used) versus July 26, 2020 (on the right, which is in the version BEERQA used).

Figure 7: Changes in Wikipedia that present challenges in matching them across years. We highlight portions of the text that have been deleted in <u>red underlined text</u>, that have been added in **green boldface text**, and that have been significantly paraphrased in *orange italics*, and leave near-verbatim text in the normal font and color.

to answer the question as a result of the mapping process.

B Elasticsearch Setup

We set up Elasticsearch in standard benchmark settings (SQuAD Open and HotpotQA fullwiki) following practices in previous work (Chen et al., 2017; Qi et al., 2019), with minor modifications to unify these approaches.

Specifically, to reduce the context size for the Transformer encoder in IRRR to avoid unnecessary computational cost, we primarily index the individual paragraphs in the English Wikipedia. To incorporate the broader context from the entire article, as was done by Chen et al. (2017), we also index the full text for each Wikipedia article to help with scoring candidate paragraphs. Each paragraph is associated with the full text of the Wikipedia article it originated from, and the search score is calculated as the summation of two parts: the similarity between query terms and the paragraph text, and the similarity between the query terms and the full text of the article.

For query-paragraph similarity, we use the standard BM25 similarity function (Robertson et al., 1994) with default hyperparameters ($k_1 = 1.2$, $b =$ 0.75). For query-article similarity, we find BM25 to be less effective, since the length of these articles overwhelm the similarity score stemming from important rare query terms, which has also been reported in the information retrieval literature (Lv and Zhai, 2011). Instead of boosting the term frequency score as considered by Lv and Zhai (2011), we extend BM25 by taking the square of the IDF term and setting the TF normalization term to zero ($b = 0$), which is similar to the TF-IDF implementation by Chen et al. (2017) that is shown effective for SQuAD Open.

Specifically, given a document D and query Q, the score is calculated as

$$\text{score}(D, Q) = \sum_{i=1}^{n} \text{IDF}_+^2(q_i) \cdot \frac{f(D, q_i) \cdot (1 + k_1)}{f(D, q_i) + k_1}, \tag{1}$$

where $\text{IDF}_+(q_i) = \max(0, \log((N - n(q_i) + 0.5)/(n(q_i) + 0.5))$, with N denoting the total number of documents and $n(q_i)$ the document frequency of query term q_i, and $f(q_i, D)$ is the term frequency of query term q_i in document D. We set $k_1 = 1.2$ in all of our experiments. Intuitively, compared to the standard BM25, this scoring function puts more emphasis on important, rare term over-

Parameter	Value
Learning rate	3×10^{-5}
Batch size	320
Iteration	10,000
Warming-up	1,000
Training tokens	1.638×10^9
Reranker Candidates	5

Table 7: Hyperparameter setting for IRRR training.

laps while it is less dampened by document length, making it ideal for an initial sift to find relevant documents for open-domain question answering.

C Further Training and Prediction Details

We include the hyperparameters used to train the IRRR model in Table 7 for reproducibility.

For our experiments using SQuAD for training, we also follow the practice of Asai et al. (2020) to include the data for SQuAD 2.0 (Rajpurkar et al., 2018) as negative examples for the reader component. Hyperparameters like the prediction threshold of binary classifiers in the query generator are chosen on the development set to optimize end-to-end QA performance.

We also include how we use the reader model's prediction to stop the IRRR pipeline for completeness. Specifically, when the most likely answer is yes or no, the answerability of the reasoning path is the difference between the yes/no logit and the `NOANSWER` logit. For reasoning paths that are not answerable, we further train the span classifiers to predict the `[CLS]` token as the "output span", and thus we also include the likelihood ratio between the best span and the `[CLS]` span if the positive answer is a span. Therefore, when the best predicted answer is a span, its answerability score is computed by including the score of the "`[CLS]` span" as well, *i.e.*,

$$\text{Answerability}_{\text{span}}(p) = \text{logit}_{\text{span}} - \text{logit}_{\text{NOANSWER}}$$
$$+ \frac{\text{logit}_s^{\text{start}} - \text{logit}_{\text{[CLS]}}^{\text{start}}}{2}$$
$$+ \frac{\text{logit}_e^{\text{end}} - \text{logit}_{\text{[CLS]}}^{\text{end}}}{2}, \quad (2)$$

where $\text{logit}_{\text{span}}$ is the logit of predicting span answers from the 4-way classifier, while $\text{logit}^{\text{start}}$ and $\text{logit}^{\text{end}}$ are logits from the span classifiers for selecting the predicted span from the reasoning path.

Question	What **team** was the **AFC champion**?
Step1 (Non-Gold)	However, the eventual-AFC Champion **Cincinnati Bengals**, playing in their first AFC Championship Game, defeated the Chargers 27-7 in what became known as the Freezer Bowl. ...
Step2 (Non-Gold)	Super Bowl XXVII was an American football game between the American Football Conference (AFC) champion **Buffalo Bills** and the National Football Conference (NFC) champion Dallas Cowboys to decide the National Football League (NFL) champion for the 1992 season. ...
Gold	Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champion **Denver Broncos** defeated the National Football Conference (NFC) champion Carolina Panthers 24-10 to earn their third Super Bowl title. ...

Figure 8: An example where there are false negative answers in Wikipedia for the question from SQuAD Open.

D Further Analyses of Model Behavior

In this section, we perform further analyses and introduce further case studies to demonstrate the behavior of the IRRR system. We start by analyzing the effect of the dynamic stopping criterion for reasoning path retrieval, then move on to the end-to-end performance and leakages in the pipeline, and end with a few examples to demonstrate typical failure modes we have identified that might point to limitations with the data.

Effect of Dynamic Stopping. We begin by studying the effect of using the answerability score as a criterion to stop the iterative retrieval, reading, and reranking process within IRRR. We compare the performance of a model with dynamic stopping to one that is forced to stop at exactly K steps of reasoning, neither more nor fewer, where $K = 1, 2, \ldots, 5$. As can be seen in Table 8, IRRR's dynamic stopping criterion based on the answerability score is very effective in achieving good end-to-end question answering performance for questions of arbitrary complexity without having to specify the complexity of questions ahead of time. On both SQuAD Open and HotpotQA, it achieves competitive, if not superior question answering performance, even without knowing the true number of gold paragraphs necessary to answer each question.

Aside from this, we note four interesting findings: (1) the performance of HotpotQA does not peak at two steps of reasoning, but instead is helped by performing a third step of retrieval for the average question; (2) for both datasets, forcing the model to retrieve more paragraphs after a point consistently hurt QA performance; (3) dynamic stopping slightly hurts QA performance on SQuAD Open compared to a fixed number of reasoning steps ($K = 1$);

Steps	SQuAD Open		HotpotQA	
	EM	F_1	EM	F_1
Dynamic	49.92	60.91	**65.74**	**78.41**
1 step	**51.07**	**61.74**	13.75	18.95
2 step	38.74	48.61	65.12	77.75
3 step	32.14	41.66	65.37	78.16
4 step	29.06	38.33	63.89	76.72
5 step	19.53	25.86	59.86	72.79

Table 8: SQuAD and HotpotQA performance using adaptive vs. fixed-length reasoning paths, as measured by answer exact match (EM) and F_1. The dynamic stopping criterion employed by IRRR achieves comparable performance to its fixed-step counterparts, without knowledge of the true number of gold paragraphs.

(4) when IRRR is allowed to select a dynamic stopping criterion for each example independently, the resulting question answering performance is better than a one-size-fits-all solution of applying the same number of reasoning steps to all examples. While the last confirms the effectiveness of our answerability-based stopping criterion, the cause behind the first three warrants further investigation. We will present further analyses to shed light on potential causes of these in the remainder of this section.

Case Study for Failure Cases. Besides model inaccuracy, one common reason for IRRR to fail at finding the correct answer provided with the datasets is the existence of false negatives (see Figure 8 for an example from SQuAD Open). We estimate that there are about 9% such cases in the HotpotQA part of the training set, and 26% in the SQuAD part of the training set.

These false negatives hurt the quality of data generation as well, especially when generating the SQuAD part of the training set. We investigate randomly selected question-context pairs in the training set and find 24% of our SQuAD training set and 13% of GRR's SQuAD training set are false negatives. This means our methods find better candidate documents but true answers in those documents become false positives. That results in worse performance for our model when it is trained with only the SQuAD part of training set as shown in Table 2.

E Three+ Hop Challenge Set Analysis

Although SQuAD Open and HotpotQA probe our model's ability on single and two-hop questions, we lacked insight into the ability of our model to

# of Documents to answer the question	3	4	5	6	7	8
# of questions	495	17	8	0	9	1

Table 9: Distribution of reasoning steps for questions in Three+ Hop Challenge Set.

Reasoning Type	%
Comparison	25.6
Bridge-Comparison	25.3
Bridge	49.1

Table 10: Reasoning types required for Three+ Hop Challenge Set.

generalize to questions that require three or more reasoning steps/hops, which is more than what our model is trained on. Therefore, we built a challenge set comprised of questions that require at least three hops of reasoning to answer (see Table 9 for a breakdown of the number of documents required to answer each question in the challenge set). While the vast majority of challenge set questions require three documents, questions that require four or more documents are also present, hence the "Three+ Hop Challenge Set" name. Although we intend to use the challenge set for testing only, we will share a few key insights into the question sourcing process, the reasoning types required, and the answer types present.

Question Sourcing Process. We annotated 530 examples that require three or more paragraphs to be answered on the 2020 Wikipedia dump. We developed roughly 50–100 question templates that cover a diverse set of topics, including science, literature, film, music, history, sports, technology, politics, and geography. We then annotated approximately ten to twenty examples from each of these question templates to ensure that the resulting challenge set contained a diverse set of topics and questions.

Reasoning Types. During the annotation process for the challenge set, we recorded the types of reasoning required to answer each question (Table 10). Roughly half of the questions require chain reasoning (Bridge), where the reader must identify bridge entities that link the question to the first context paragraph, the first context paragraph to the second, and finally the second to the third where the answer can be found. In the case

Answer Type	%	Example(s)
Person	29	Kate Elizabeth Winslet
Number	20	388,072, 5.5 million
Yes / No	15	—
Group / Org	11	CNN
Date	8	March 28, 1930
Other Proper Noun	7	Boeing 747-400
Creative Work	5	"California Dreams"
Location	4	New York City
Common Noun	1	comedy-drama

Table 11: Types of answers in Three+ Hop Challenge Set. These statistics are based on 100 randomly sampled examples.

that four or more hops of reasoning are required, this chain of reasoning will extend past the third paragraph to the n-th paragraph where the answer can be found. Additionally, approximately 25% of the questions require the comparison of three or more entities (Comparison). For these questions, the reader needs to retrieve three or more context paragraphs identified in the question that are not directly connected to each other and then compare them on certain aspects specified in the question, similar to the comparison questions in HotpotQA. The remaining 25% of the questions require both chain reasoning and the comparison of two or more entities (Bridge-Comparison). For these questions, the reader must first identify a bridge entity that links the question to the first context paragraph. They then must identify two or more entities to compare within the first context paragraph. Afterwards, they retrieve context paragraphs for each of the aforementioned entities and compare them on certain aspects specified in the question.

Answer Types. We also analyze the types of answers present in the challenge set. As shown in Table 11, the challenge set features a diverse set of answers. We find that roughly half of the questions ask about people (29%) and numeric quantities (20%). Additionally, we find a considerable number of questions that require a yes or no answer (15%), ask about groups or organizations (11%), dates (8%), and other proper nouns (7%). The challenge set also contains a non-negligible amount of questions that ask about creative works (5%), locations (4%), and common nouns (1%).