Context in Question Answering

James Ferguson

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington
2022

Reading Committee:
Hannaneh Hajishirzi, Chair
Noah Smith
Pradeep Dasigi

Program Authorized to Offer Degree:
Computer Science and Engineering

University of Washington

Abstract

Context in Question Answering

James Ferguson

Chair of the Supervisory Committee:
Associate Professor Hannaneh Hajishirzi
Computer Science and Engineering

Question Answering (QA) has seen a massive surge in interest, and a correlated improvement in model performance, over the past half decade, even surpassing humans on some datasets. Many models for this task rely on some form of contextual information that leads from the question to the answer. This context plays a crucial role in how well models perform. Using the right context when answering a question can be the difference between a model achieving superhuman performance, or performing little better than random. When constructing a dataset, using context incorrectly can trivialize the problem, resulting in models that do not generalize to other data. Seeing alternate contexts during training can help improve model robustness by providing multiple views of how to answer a question.

We present three works that explore the significance of context in QA. First, we introduce a new dataset, IIRC, in which questions that require multi-hop, discrete reasoning are written with access to only partial context. Answers are then collected separately, resulting in much lower lexical overlap between questions and relevant context. We show that these questions are easily answerable by humans, but state-of-the-art models struggle to achieve comparable results. Next, we present a new method for selecting training context. Many questions have multiple contexts that lead to the correct answer, but exhaustively annotating these contexts is difficult. We use the downstream QA loss to identify alternate contexts during training, and show that this approach enables identifying relevant context for unseen data greater than 90% of the time on

the IIRC dataset. Finally, we introduce QAPaC, Question-Answer Pairs As Context. In this approach, we segment documents into QA pairs, and then retrieve context from the collection of QA pairs. We show that this approach results in a 1.8 F1 point improvement compared to segmenting documents into either sentences or windows on IIRC. Additionally, we show that using an ensemble of QA-pairs and sentences results in a further improvement of 1.3 F1 points.

Acknowledgements

I would like to start off by extending my sincerest thanks to my advisor, Hanna Hajishirzi, for all of her guidance and support over the years. Her wisdom and teaching has moulded me into the researcher I am now. Beyond that, she helped me push through the periods in which I lacked results and motivation. Without her, I would not be where I am today.

I would next like to thank my committee members, Noah Smith, Pradeep Dasigi, and Gina-Anne Levow for their discussion and feedback, as well as bearing with my habit of scheduling things at the last minute.

I would also like to thank my collaborators over the years, including Dan Weld, Janara Christensen, Scott Yih, Matt Gardner, and Tushar Khot, for their discussions and wisdom, as well as their patience in pruning my writing into an intelligible form. I would especially like to thank Pradeep Dasigi for the mentorship and time he has given me over the past couple of years.

Penultimately, I would like to thank the members of H2Lab, and the larger UW NLP community, for fostering an open, friendly environment. I would especially like to extend my gratitude to Rik Koncel-Kedziorski, Sachin Mehta, Colin Lockard, and Aida Amini for their companionship throughout my time at UW.

Finally, I would like to thank my parents, Robert and Dana, and my brother, Chris, for being an unending font of support throughout my life, for feigning understanding even when I depart on infinite tangents when explaining my work, and for stimulating a love of science and technology that has ultimately led me to this point.

DEDICATION

To my parents

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Asking questions has always been a way of checking understanding, whether it be asking for clarification in a casual conversation or evaluating language understanding in a more formal setting like a school exam. In natural language processing, question answering (QA) plays the same role, serving as a benchmark for evaluating a system's ability to understand and reason about language. Specifically, a system is given some textual context, and one or more questions that can be answered using the information in that context. In its basic form, that of reading comprehension, this context is limited to a paragraph or two, allowing the system to process the entire context all at once when answering the question. In a more challenging form, known as open-domain QA, the context is a much larger collection of documents, such as all of Wikipedia[1], and the model must first retrieve relevant snippets of text before computing an answer. This is the formulation that we will be focusing on in this dissertation.

Each of these cases has access to different kinds of context, and each comes with its own set of challenges, both in modeling decisions, as well as during the data collection process. Typically for the reading comprehension questions, the entire context can be encoded all at once, so the bulk of the modeling difficulty comes from selecting an answer from the context. Meanwhile, in open-domain QA, relevant context has to be selected from a much larger collection of information, so modeling that problem also requires efficiently searching over a large space to identify relevant information before attempting to answer the question.

During data collection each of these problems also comes with unique challenges. In the reading

[1] www.wikipedia.org

comprehension setting, annotators are typically given access to the passage of text and asked to generate question-answer pairs answerable by that text. This often results in biases in which there is significant lexical overlap between the questions and the context surrounding the answer [Jia and Liang, 2017]. On the other hand, collecting questions in the open-domain setting requires annotators to manually go through the process of searching a large collection of documents in order to identify relevant context. Because of the large size of this collection, exhaustively annotating all relevant context is intractable, resulting in the existence of false negatives in the training data.

In this work, we primarily focus on the open-domain setting. We first introduce IIRC, a dataset that contains questions with a variety of reasoning types, including discrete and multi-context reasoning. We then propose an approach to mitigate the existence of false negatives by automatically identifying them and using them for data augmentation as additional positive labels. Finally, we introduce a novel method of representing context as question-answer pairs, providing the retrieval model with less noisy targets during retrieval, resulting in improved performance.

1.1 Dissertation Outline

In chapter 2 we provide a formal definition of question answering problem. We then cover a brief history of different QA approaches and datasets, along with some discussion of contemporary related work.

We then introduce a new dataset of **i**ncomplete **i**nformation **r**eading **c**omprehension questions, IIRC, in chapter 3. IIRC was specifically designed to address some of the shortcomings of previous datasets. Questions in IIRC are more difficult, requiring models to identify what information is missing from a passage, then perform discrete reasoning over multiple contexts to find the answer. Additionally, questions were collected by annotators without access to the full context, resulting in naturally unanswerable questions, as well as minimal token overlap between the question and relevant context.

In chapter 4, we next propose a novel approach for retrieval data augmentation based on downstream QA performance. Due to the difficulty of exhaustively annotating relevant context from large collections of data, there are many false negatives, or alternate contexts that also contain enough information to answer a question, but which are not labeled as relevant in the dataset. Building on the intuition that a retrieval model should identify context that, when passed to a QA model, has a high probability of correctly answering the

question, we rank potential contexts based on downstream QA model scores. We show that augmenting training data with contexts selected according to this score results in improved performance both for retrieval, and downstream QA.

In chapter 5, we present **Q**uestion-**A**nswer **P**airs **a**s **C**ontext (QAPaC), which is an approach for representing context documents as a collection of generated question-answer pairs, reducing the noise that is introduced with more static document segmentation approaches. This builds on the idea that relevant information is often expressed in short clauses, and we can use generated QA pairs to target specific pieces of information compared to retrieving sentences or fixed-size windows of text. We show that segmenting documents in this way results in improved retrieval performance, which also leads to better downstream performance.

Finally, in chapter 6 we summarize the contributions, and briefly discuss future directions for this work.

1.2 Publications

Parts of this work appear in the following publications:

James Ferguson, Matt Gardner, Hannaneh Hajishirzi, Tushar Khot, and Pradeep Dasigi. 2020. IIRC: A dataset of incomplete information reading comprehension questions. In *ACL*.

James Ferguson, Hannaneh Hajishirzi, Pradeep Dasigi, and Tushar Khot. 2022. Retrieval data augmentation informed by downstream question answering performance. In *Proceedings of the Fifth Fact Extraction and VERification Workshop (FEVER)*, pages 1–5, Dublin, Ireland. Association for Computational Linguistics.

Chapter 2

Background

In section 2.1 we provide a formal definition of the general QA problem. Then in section 2.2 we briefly discuss the history of question answering, covering approaches to answering questions conditioned on context (Section 2.2.1), QA datasets (Section 2.2.2), and approaches for context retrieval (Section 2.2.3).

2.1 Problem Definition

In this section, we define some terminology that we will be using to describe question answering. In this work, we focus on multi-context, open-domain question answering. This means that questions typically require information from multiple documents, and those documents are drawn from a large corpus of candidates.

Our approach uses the standard two-step pipeline for open-domain QA seen in prior work. We first run a retrieval model that takes as input a question, q, and the large corpus of documents, C, and outputs a small subset of passages from those documents, $c \subset C$, that contains sufficient information to answer the question.

In order to efficiently identify a subset of passages, we first segment the documents in C. In chapters 3 and 4 we do this by splitting the documents into sentences. In chapter 5 we explore alternative methods for splitting documents. We also simplify the subset selection problem by scoring each passage independently, and returning the top k.

The resulting subset of passages is then passed to the second step: the QA model. This model takes as input the same question, q, and subset of passages, c, from the first step, and outputs an answer, a. Depending on the data, this answer can take many forms, such as a span from the context, a number, yes/no, or none of

these if the question is unanswerable.

2.2 Related Work

2.2.1 Question Answering

When question answering was first introduced, it was done by mapping English words to specific predicates/-code. These systems did not take additional context as input, but were designed for very specific domains, such as baseball [Green et al., 1961] or weather [Woods, 1977], more akin to querying a database via natural language. These systems relied on manually constructed mappings of words to functions and predicates that were then run on the underlying knowledge source, resulting in an answer. As computers progressed, and data became more readily available, systems began to make use of context. Systems such as Quarc [Hirschman et al., 1999] and Deep read [Riloff and Thelen, 2000] relied on word overlap to both identify relevant context and select answer sentences from within that context. Progressing beyond simple word overlap, Harabagiu et al. [2000] proposed instead comparing semantic representations of the question and context to identify answers. They still select relevant context based on word overlap, but then parse both the question and possible context into structured semantic representations. They then use overlaps in the semantic structure, along with a heuristically predicted answer type, to select the answer.

Also around this time, simple machine learning approaches began to emerge, such as that proposed by Ng et al. [2000]. Limited by both the size of the datasets and the computational power available, they use a logistic regression model with hand-crafted features as input. They select constituents from a constituency parser as potential answer spans, and then use the logistic regression model to label each one. This general approach remained prevalent until the introduction of better computers and larger datasets, such as SQuAD [Rajpurkar et al., 2016], which allowed for more powerful models, such as neural networks, to be used. Around this time, approaches such as Bordes et al. [2014]; Iyyer et al. [2014] propose using neural networks for question answering. Bordes et al. [2014] use Freebase to provide candidate answers, and learn embeddings for questions and Freebase subgraphs. They then score candidate answers by based on the dot product of their embeddings with that of the question. Iyyer et al. [2014] combine syntactic structure and neural networks using a dependency-tree recurrent neural network. They first compute a dependency parse of the question,

and then recursively work up the tree, computing representations for each node as a function of its children. The root embedding is then used to score each answer candidate.

The next major leap involved encoding question and context together to predict answers [Hermann et al., 2015; Seo et al., 2017]. Seo et al. [2017] encode both question and context using separate LSTMs, but with attention over the other. This creates a context-aware encoding of the question, and similarly, a question-aware encoding of the context, allowing each encoding to focus on relevant information conditioned on the other. Hermann et al. [2015] propose multiple models using long short-term memory networks (LSTMs) that take as input the question concatenated with context to compute a representation of both together. Now, with the advent of large, pretrained language models (LMs) [Devlin et al., 2019; Liu et al., 2019], this approach of encoding the concatenated question and context is widespread.

Additionally, as language models grow larger and more powerful, there has been increasing interest in using generation models for QA that generate answers conditioned on the context instead of extracting spans. These models have the advantage of occasionally being able to correctly predict answers even in cases where the retrieval model fails to identify the correct context. UnifiedQA [Khashabi et al., 2020] uses a single generation framework to combine data from a variety of QA datasets in order to improve generality. RAG [Lewis et al., 2020b] and REALM [Guu et al., 2020a] train generation models augmented with information retrieved from a large indexed text corpus. More recently, Brown et al. [2020] show that language models can be used directly for solving various NLP tasks, including QA, with minimal finetuning. They show that by selecting the right input structure, large, pretrained language models can obtain results comparable to models finetuned for specific tasks. This has led to considerable work in searching over the input space, known as *prompt-tuning*, and has achieved state-of-the-art results on many QA datasets in the few-shot setting. Li and Liang [2021] and Lester et al. [2021] propose tuning "soft prompts" by freezing the model parameters, and instead learning task-specific vectors.

Current models have been able to surpass human performance even on larger datasets, like SQuAD, leading some to explore alternate formulations of the question answering problem. This has resulted in datasets covering more difficult types of reasoning, such as science [Clark et al., 2018], math [Amini et al., 2019; Dua et al., 2019b], and multi-hop reasoning chains [Yang et al., 2018]. Models for these domains generally require some form of additional processing in order to reach the answer. For example, in the math

domain, Yu et al. [2021] first encode the question, then generate an equation tree, which is then passed to a solver to compute a final numeric answer. in the multi-hop domain, the additional processing is typically required at retrieval time as well. Min et al. [2019c] first decomposes the question into multiple subquestions. It then attempts to retrieve context and answer each subquestion independently, and uses the results to compute the answer to the original question.

2.2.2 Datasets

The early question answering systems were designed with hand-crafted rules for very specific domains, and thus did not require any kind of learning, and so did not need any data. Early models with probabilistic components, such as parsers, relied on other datasets to train those component, such as the Penn Treebank [Marcus et al., 1993] for parsers, in order to train. The QA portion of these models simply relied on matching up portions of the outputs of these components. As such, datasets were primarily used for evaluation rather than learning model parameters. For example, TREC-8 [Voorhees and Tice, 1999] consists of only a 200 question test set. Meanwhile, the data used by early machine learning approaches, such as Ng et al. [2000], consists of a training set of approximately 300 questions.

As machine learning became more prevalent, datasets began to grow in size. At this time, datasets generally fell into one of two categories. The first were large collections of primarily synthetic data. For example, the CNN/Daily Mail dataset [Hermann et al., 2015] is a collection of articles paired with human annotated summaries. One entity is masked from each sentence in the summary, resulting in over a million cloze-style questions. The second kind of dataset at this time had high-quality manually collected data, but was smaller in size. One popular datasete, MCTest [Richardson et al., 2013], consists of 2000 manually written questions for 500 stories. Meanwhile, another, ProcessBank [Berant et al., 2014] has only 585 expert-written questions about biological processes.

This continued until the emergence of SQuAD [Rajpurkar et al., 2016], which is a large crowdsourced reading comprehension dataset. SQuAD consists of upwards of 100K questions concerning Wikipedia articles, generated by crowd workers. While it is not without its problems, as discussed above, this allowed for the development of new large machine learning models, and opened the door for a large number of modern datasets spread across a variety of QA tasks.

Questions requiring multiple contexts Prior multi-context reading comprehension datasets were built by starting from discontiguous contexts, and forming compositional questions by stringing multiple facts either by relying on knowledge graphs as in QAngaroo [Welbl et al., 2018], or by having crowdworkers do so, as in HotpotQA [Yang et al., 2018]. It has been shown that many of these questions can be answered by focusing on just one of the facts used for building the questions [Min et al., 2019b]. In contrast, our work, IIRC, contains questions written by a crowdworker who had access to just one paragraph, with the goal of obtaining information missing in it, thus minimizing lexical overlap between questions and the answer contexts. Additionally, IIRC provides a unique question type: questions requiring aggregating information from many related documents, such as the second question in Figure 3.2.

Separation of questions from answer contexts One of the main problems with questions in SQuAD is the lexical overlap between questions and the context immediately surrounding the answer. A multitude of datasets (e.g.: WhoDidWhat [Onishi et al., 2016], NewsQA [Trischler et al., 2016], DuoRC [Saha et al., 2018], Natural Questions [Kwiatkowski et al., 2019], TyDiQA [Clark et al., 2020]) have tried to address this by separating the contexts that questions are anchored in from those that are used to answer them. IIRC also separates the two contexts, but is unique given that the linked documents elaborate on the information present in the original contexts, naturally giving rise to follow-up questions, instead of open-ended ones.

Open-domain question answering In the open-domain QA setting, a system is given a question without any associated context, and must retrieve the necessary context to answer the question [Chen et al., 2017; Joshi et al., 2017; Dhingra et al., 2017; Yang et al., 2018; Seo et al., 2019; Karpukhin et al., 2020; Min et al., 2019a]. IIRC is similar in that it also requires the retrieval of missing information. However, the questions are grounded in a given paragraph, meaning that a system must examine more than just the question in order to know what to retrieve. Most questions in IIRC do not make sense in an open-domain setting, without their associated paragraphs.

Unanswerable questions Unlike SQuAD 2.0 [Rajpurkar et al., 2018] where the unanswerable questions were written to be close to answerable questions, IIRC contains naturally unanswerable questions that were not written with the goal of being unanswerable, a property that our dataset shares with NewsQA [Trischler

et al., 2016], Natural Questions [Kwiatkowski et al., 2019], and TyDi QA [Clark et al., 2020]. Results shown in Section 3.4.3 indicate that these questions cannot be trivially distinguished from answerable questions.

Incomplete Information QA A few other datasets have explored question answering given incomplete information, such as science facts [Mihaylov et al., 2018; Khot et al., 2019b]. However, these datasets contain multiple choice questions, and the answer choices provide hints as to what information may be needed. Yuan et al. [2020] explore this as well using a POMDP in which the context in existing QA datasets is hidden from the model until it explicitly searches for it.

2.2.3 Retrieval

Early retrieval models relied on word overlap between question and potential contexts. Moldovan et al. [1999] first preprocess the question by identifying keywords, and then scoring context based on heuristics considering which of the keywords appear. One step up from that are models using TF-IDF and BM25 [Robertson and Zaragoza, 2009a]. Both approaches also rely on token overlap, but weight tokens according to the number of times they appear in a document and the inverse of the number of documents they appear across the entire corpus. Although these approaches still rely on word overlap, they remain competitive baselines that can be run very efficiently on large collections of documents without requiring any training.

More recently, Karpukhin et al. [2020] proposed dense passage retrieval (DPR), and showed that a retrieval model could be learned and run efficiently enough to work on large corpora found in open-domain QA. They split documents into segments of 100 tokens using a sliding window, and then index each segment offline. They then use a dual-encoder and score each segment by taking a dot product with the question encoding. Because not all datasets have annotated retrieval contexts, they rely on an answer-span heuristic to identify relevant contexts for trainng the model. Specifically, they label a segment as a positive retrieval target if the answer to the question appears in that segment. While this is obviously a noisy heuristic, it has proven quite effective for many datasets. This has resulted in a large amount of work exploring new ways of training retrieval models.

Lee et al. [2019] pretrain a retrieval model using an inverse cloze task. Zhao et al. [2021] more recently proposed to iteratively improve a retrieval model using hard-EM. Both approaches filter the data using the answer span heuristic. This heuristic breaks down on multi-hop questions, as well as questions that are

not answerable by spans, such as true/false or discrete reasoning questions. Izacard and Grave [2021] and Yang and Seo [2021] propose using knowledge distillation to incorporate QA information into a supervised retriever, and while assuming access to retrieval annotations, Ni et al. [2020] jointly learn retrieval and QA by marginalizing over potential contexts. All three of these approaches require encoding all potential contexts together with the question, which is memory intensive, limiting the number of contexts the models are able to consider.

Chapter 3

IIRC: A Dataset of Incomplete Information Reading Comprehension Questions

3.1 Introduction

Humans often read text with the goal of obtaining information. Given that a single document is unlikely to contain all the information a reader might need, the reading process frequently involves identifying the information present in the given document, and what is missing, followed by locating a different source that could potentially contain the missing information. Most recent reading comprehension tasks, such as SQuAD 2.0 [Rajpurkar et al., 2018], DROP [Dua et al., 2019b], or Quoref [Dasigi et al., 2019], evaluate models using a relatively simpler setup where all the information required to answer the questions (including judging them as being unanswerable) is provided in the associated contexts. While this setup has led to significant advances in reading comprehension [Ran et al., 2019; Zhang et al., 2020], the tasks are still limited since they do not evaluate the capability of models at identifying precisely what information, if any, is missing to answer a question, and where that information might be found.

On the other hand, open-domain question answering tasks [Chen et al., 2017; Joshi et al., 2017; Dhingra et al., 2017] present a model with a question by itself, requiring the model to retrieve relevant information from some corpus. However, this approach loses grounding in a particular passage of text, and it has so far been challenging to collect diverse, complex question in this setting.

Alternatively, complex questions grounded in context can be converted to open-domain or incomplete-information QA datasets such as HotpotQA [Yang et al., 2018]. However, they do not capture the information-seeking questions that arise from reading a single document with partial information [Min et al., 2019b; Chen and Durrett, 2019].

We present a new dataset of incomplete information reading comprehension questions, IIRC, to address both of these limitations. IIRC is a crowdsourced dataset of 13441 questions over 5698 paragraphs from English Wikipedia, with most of the questions requiring information from one or more documents hyperlinked to the associated paragraphs, in addition to the original paragraphs themselves. Our crowdsourcing process (Section 3.2) ensures the questions are naturally information-seeking by decoupling question and answer collection pipelines. Crowd workers are instructed to ask follow-up questions after reading a paragraph, giving links to pages where they would expect to find the answer. This process results in questions like the one shown in Figure 3.1. As illustrated by the example, this setup results in questions requiring complex reasoning, with an estimated 39% of the questions in IIRC requiring discrete reasoning. Moreover, 30% of the questions in IIRC require more than one linked document in addition to the original paragraph and 30% of them are unanswerable even given the additional information. When present, the answers are either extracted spans, boolean, or values resulting from numerical operations.

To evaluate the quality of the data, we run experiments with a modified version of NumNet+ [Ran et al., 2019], a state-of-the-art model from DROP [Dua et al., 2019b], chosen because a significant portion of questions in IIRC require numerical reasoning similar to that found in DROP. Because DROP uses only a single paragraph of context, we add a two-stage pipeline to retrieve necessary context for the model from the linked articles. The pipeline first identifies which links are pertinent, and then selects the most relevant passage from each of those links, concatenating them to serve as input for the model (Section 3.3).

This baseline achieves an F_1 score of 31.1% on IIRC, while the estimated human performance is 88.4% F_1. Even giving the model oracle pipeline components results in a performance of only 70.3%. Taken together, these results show that substantial modeling is needed both to identify and retrieve missing information, and to combine the retrieved information to answer the question (Section 3.4). We additionally perform qualitative analysis of the data, and find that the errors of the baseline model are evenly split between retrieving incorrect information, identifying unanswerable questions, and successfully reasoning over the retrieved information.

By construction, all examples in IIRC require identifying missing information. Even though current model performance is quite low, a model trained on this data could theoretically leverage that fact to achieve artificially high performance on test data, because it does not have to first determine *whether* more information is needed. To account for this issue, we additionally sample questions from SQuAD 2.0 [Rajpurkar et al., 2018] and DROP [Dua et al., 2019b], which have similar question language to what is in IIRC, putting forward this kind of combined evaluation as a challenging benchmark for the community. Predictably, our baseline model performs substantially worse in this setting, reaching only 28% F_1 on the IIRC portion of this combined evaluation (Section 3.5).

3.2 Building IIRC

We used Wikipedia to build IIRC and relied on the fact that entities in Wikipedia articles are linked to other articles about those entities, providing more information about them. Our goals were to build a dataset with naturally information-seeking questions anchored in paragraphs with incomplete information, such that identifying the location of missing information is non-trivial, and answering the questions would require complex cross-document reasoning.

We ensured that the questions are information-seeking by separating question and answer collection processes, and by not providing the question writers access to the contexts where the answers occur. This process also ensured that we get questions that have minimal lexical overlap with the answer contexts. We used Wikipedia paragraphs with many outgoing links to increase the difficulty of identifying the articles that provide the missing information. To ensure complex cross-document reasoning, we asked the crowd workers to create questions that need information from the seed paragraph as well as one or more linked articles. This constraint resulted in questions that are answerable neither from the original paragraph alone, nor from one of the linked articles alone, often requiring over 3+ passages to answer. The remainder of this section describes our data collection process.

3.2.1 Seed Paragraphs

We started by collecting paragraphs from Wikipedia articles containing ten or more links to other Wikipedia articles. This resulted in roughly 130K passages. We then created two separate crowdsourcing tasks on

	Type	Description	Percentage
Link Prediction	Easy	Link is explicitly mentioned in the question	41%
	Medium	Context is required to determine link target	47%
	Hard	Context is required to determine link targets and number of links	12%
Retrieval	Linked context only	Original passage is not necessary to answer question	14%
	Bridge	Original passage is only necessary to determine links	57%
	Cross context	Original passage is necessary to find relevant information in links	29%
Reasoning	Non-discrete	No discrete reasoning is required	61%
	Discrete-numeric	Discrete reasoning is required	11%
	Discrete-temporal	Discrete reasoning involving time is required	28%
Answer	Span	Answer is one or more spans selected from question or context	45%
	Numeric	Answer is a number (with a unit provided)	17%
	Binary	Answer is either *yes* or *no*	8%
	None	Question cannot be answered given the provided context	30%

Table 3.1: Frequency of different types of retrieval, reasoning, and answers that appear in IIRC.

Amazon Mechanical Turk[1]; one for collecting questions, and one for collecting answers. Workers for each task were chosen based on a qualification task. Their submissions were manually inspected, and those that produced high quality questions and correct answers, respectively, continued to work on the main annotation tasks.

3.2.2 Collecting Questions

Given a paragraph with links to other articles highlighted, crowd workers were tasked with writing questions that require information from the paragraph, as well as from one or more of linked articles. Workers could see the links, and the titles of the articles they pointed to, but not the contents of the linked articles. Since the linked articles were not provided, the workers were asked to frame questions based on the information they think would be contained in the those articles. For each human intelligence task (HIT), workers were presented with a collection of ten paragraphs, and were asked to write a total of ten questions using any of those paragraphs, with two questions requiring following two or more links. For example given a passage about an actor that mentions *Rasulala had roles in Cool Breeze (1972), Blacula (1972), and Willie Dynamite (1973)*, an example of a question requiring multiple links would be *How many different directors did Rasulala work with in 1972?*.

[1] www.mturk.com

In order to minimize questions with shortcut reasoning, we provided workers extensive instructions along with examples of good and bad questions to ask. Examples of bad questions included questions that did not require any links - *Who did the Arakanese kings compare themselves to?* when the context included *They compared themselves to Sultans*; and questions that did not require information from the original passage - *What was Syed Alaol's most famous work?* when the context included *Syed Alaol was a renowned poet*.

In addition to writing questions, workers also provided the context from the original paragraph that they thought would be necessary to answer the question, as well as the links they expected to contain the remaining necessary information. Workers were paid $\$4.00$ per set of ten questions, and reported taking 25 minutes on average, coming out to $\$9.60$ per hour. 40 workers passed the qualification and worked on the main task.

3.2.3 Collecting Answers

For the answer task, workers were given a collection of ten questions, their respective original paragraphs, and the context/links selected by the question writer. For each paragraph, workers were able to see the links, and could follow them to view the text, not including tables or images, of the linked document.

They were then asked to select an answer from one of four types: a span of text from either the question or a document, a number and unit, yes/no, or *no answer*. For answerable questions, i.e. any of the first three types, they were additionally asked to provide the minimal context span(s), necessary to answer the question. For unanswerable questions, there is typically no indication that the answer is not given, so no such context can be provided. For example, the following question was written for a passage about a ship called the Italia: *Who was the mayor of New York City when Italia was transferred to Genoa-NYC?* Following the link to New York City mentions the current mayor, but not past mayors, making it unanswerable.

Annotators were also given the option of labeling a question as bad if it didn't make sense, and these bad questions were then filtered out. For example, if an annotator misinterpreted the passage when writing the question as in the case of the following question written about a horse, Crystal Ocean, and St Leger, which the annotator thought was a horse, but is actually a horse race: *Is Crystal Ocean taller than St Leger?*. Additionally, A small percentage of questions that can be answered from the original paragraph alone were also marked as being bad.

For the training set, comprising 80% of the data, each question was answered by a single annotator. For

Number of questions	13441
Number of passages	5698
Average number of links per passage	14.5
Average passage length (words)	197.5
Average question length (words)	13.6

Table 3.2: Statistics of IIRC.

the development and test sets, comprising 10% each, three annotators answered each question, and only questions where at least two annotators agreed on the answer were kept. Workers were paid $3.00 per set of ten answers, and reported taking 20 minutes on average, coming out to $9.00 per hour. 33 workers passed the qualification and worked on the main task.

3.2.4 Dataset Analysis

In Figure 3.2 we show some examples from IIRC, labeled with different kinds of processing required to solve them. The types are described in detail in Table 3.1. These types and percentages were computed from a manual analysis of 100 examples.

In Table 3.2 we provide some global statistics of the dataset. In total, there are 13441 questions over 5698 passages. Each passage contains an average of 14.5 outgoing links. Using the context provided by the answer annotators, we are able to compute a distribution of the number of links required to answer questions in the dataset, included in Table 3.4. While the majority of questions require information from only one linked document in addition to the original paragraph, 30% of questions require two or more, with some requiring reasoning over as many as 12 documents to reach the answer. This variability in the number of context documents adds an extra layer of complexity to the task.

We also analyzed the initial trigrams of questions to quantify the diversity of questions in the dataset. We found that the most common type of questions, those related to time (eg "How old was", "How long did"), make up 15% of questions. There are 3.5k different initial trigrams across the 10.8k questions in the training set.

3.3 Modeling IIRC

3.3.1 Task Overview

Formally, a system tackling IIRC is provided with the following inputs: a question Q; a passage P ; a set of links contained in the passage, $L = \{l_i\}_{i=1}^{N}$; and the set of articles those links lead to, $\mathbf{A} = \{a^i\}_{i=1}^{N}$. The surface form of each link, l_i is a sequence of tokens in P and is linked to an article a^i. The target output is either a number, a sequence of tokens in one of P, Q, or a^i, `Yes`, `No`, or `NULL` (for unanswerable questions).

3.3.2 Baseline Model

To evaluate the difficulty of IIRC, we construct a baseline model adapted from a state-of-the-art model built for DROP. We choose a DROP model due to the inclusion of numerical reasoning questions in our dataset. Because the model was not originally used for data requiring multiple paragraphs and retrieval, we first predict relevant context to serve as input to the QA model using a pipeline with three stages:

1. Identify relevant links
2. Select passages from linked articles
3. Pass the concatenated passages to a QA model

Identifying Links

To identify the set of relevant links, L′, in a passage, P, for a question, Q, the model first encodes the concatenation of the question and original passage using BERT [Devlin et al., 2019]. It then concatenates the encoded representations of the first and last tokens of each link as input to a scoring function, following the span classification procedure used by Joshi et al. [2020], selecting any links that score above a threshold g.

$$P' = \text{BERT}([Q||P])$$

$$\text{Score(l)} = f([p'_i \| p'_j]), \;\; l = (p_i ... p_j, a)$$

$$L' = \{l : \text{Score}(l) > g\}$$

where l is a link covering tokens $p_i ... p_j$ linking to article a.

Selecting Context

Given the set, L' from the previous step, the model then must select relevant context passages from the documents. For each document, it first splits the document into overlapping windows[2], $w_0, w_1 ... w_n$. Each window is then concatenated with the question and prepended with a `CLS` token, and encoded with BERT. The encoded `CLS` tokens are then passed through a linear predictor to score each window, and the highest scoring sections from each document are concatenated as context for the final model, C.

$$c_{a_i} = \max_{w_j \in \text{Split}(a_i)} f(\text{BERT}([Q||w_j]))$$

$$C = [c_{a_i} : a_i \in L']$$

QA Model

As mentioned above, the final step in the pipeline is passing the concatenated context, along with the question and a selected window from the original passage, as input to a QA model. For our experiments, we use NumNet+, because it is the best performing model on the DROP leaderboard with publicly available code. At a high level, NumNet+ encodes the input using RoBERTa [Liu et al., 2019], as well as a numerical reasoning component. It then passes these into a classifier to determine the type of answer expected by the question, which we modified by adding binary and unanswerable as additional answer types. This model is trained using the gold context for answerable questions, and predicted context for unanswerable questions. We do this because by definition, unanswerable questions do not have annotated answer context.

3.4 Experiments

3.4.1 Evaluation Metrics

We use two evaluation metrics to compare model performance: Exact-Match (EM), and a numeracy-focused (macro-averaged) F1 score, which measures overlap between a bag-of-words representation of the gold and predicted answers. Due to the number of numeric answers in the data, we follow the evaluation methods used by DROP [Dua et al., 2019b].

[2]See section 3.4.2 for more details.

Specifically, we employ the same implementation of Exact-Match accuracy as used by SQuAD [Rajpurkar et al., 2016], which removes articles and does other simple normalization, and our F1 score is based on that used by SQuAD. We define F1 to be 0 when there is a number mismatch between the gold and predicted answers, regardless of other word overlap. When an answer has multiple spans, we first perform a one-to-one alignment greedily based on bag-of-word overlap on the set of spans and then compute average F1 over each span. For numeric answers, we ignore the units. Binary and unanswerable questions are both treated as span questions. In the unanswerable case, the answer is a special `NONE` token, and in the binary case, the answer is either *yes* or *no*.

3.4.2 Implementation Details

For the link selection model, we initialized the encoder with pretrained BERT-base, and fine-tuned it during training. For the scoring function, we used a single linear layer with a sigmoid activation function. The model was trained using Adam, and the score threshold to select links was set to 0.5. Additionally, we truncated any passages longer than 512 tokens to 512. This occurred in less than 1% of the data. This model is trained using a cross-entropy objective with the information provided in the gold context by annotators. Any links pointing to articles with an annotated context span are labeled 1, and all other links are labeled 0.

For the passage selection model, we again initialized the encoder with pretrained BERT-base, and fine-tuned it during training. We set the window size such that the concatenation of all selected contexts, along with the question and a selection from the original passage, has max length 512. More specifically, using the number of links, N_l selected in the previous step, for a question with N_Q tokens, we set the window size to be $\frac{512-(N_Q)}{N_l+1}$. We set the stride to be $\frac{1}{4}$ the window size, i.e. if the first window contains tokens $[0, 200]$, the second window would contain $[50, 250]$. We used a single linear layer with a sigmoid activation as the scoring function. We train this model with a cross-entropy objective. We use the gold context provided by annotators, labeling sections that contain the entirety of the annotated context 1, and all other sections 0.

For NumNet+, we followed the hyperparameter and training settings specified in the original paper [Ran et al., 2019]. We trained the model on gold context provided by annotators when available, i.e. for answerable questions, and predicted context from the previous steps otherwise.

	Dev		Test	
Model	**EM**	**F1**	**EM**	**F1**
Full model	29.6	33.0	27.7	31.1
Oracle L	30.9	34.7	29.0	32.5
Oracle L+C	63.9	69.2	65.6	70.3
Human	-	-	85.7	88.4

Table 3.3: Baseline and oracle results on IIRC. Human evaluation was obtained from a subset of 200 examples from the test set. We evaluate the model when given oracle links (L) and retrieved contexts (C). Retrieving the correct contexts is a significant challenge, but even given oracle contexts there is a substantial gap between model and human performance.

Number of links	**EM**	**F1**
1 (70%)	33.2	36.7
2 (23%)	27.0	30.6
3 (4%)	25.9	31.5
4+ (3%)	40.9	43.4

Table 3.4: Exact match and F1 of the baseline model on the IIRC dev set broken down by number of links necessary to answer the question. The numbers in parentheses are the percentage of questions in the full dataset that require that number of context documents.

3.4.3 Results and Discussion

Full Task Results Table 3.3 presents the performance of the baseline model. It additionally shows the results of using gold information at each stage of the pipeline, as well as human performance on computed on a subset of 200 examples from the test set. The model achieves 31.1% F_1, which is well below the human performance of 88.4%. Even with the benefit of the gold input, there is still room for improvement on reasoning over multiple contexts, as performance is still 18% absolute below human levels. The model does a good job of predicting the relevant links, as evidenced by the fact that using the gold links only improves performance by 1 point, but still struggles to identify the appropriate context within the linked documents. This is likely due to annotators not being able to see the linked context when the questions are written. This makes this step more difficult by not providing the model with surface-level lexical cues in the question that it could use to easily select the appropriate context.

Analysis of Number of Linked Documents Table 3.4 shows the results of running the full pipeline broken down according to the number of linked documents required to answer the question. These performance

Answer Type	EM	F1
Span	24.0	29.1
Number	20.4	-
Binary	56.5	-
No Answer	32.4	-

Table 3.5: Exact match and F1 of the baseline model on the IIRC dev set broken down by answer type. F1 equals EM for non-span types, so is not repeated.

Input	P	R	F1
Constant baseline	26.7	100.0	42.1
Question only	61.8	54.9	58.1
Question + Passage	64.2	54.9	59.2
Question + Pred Context	62.3	70.1	66.0

Table 3.6: Precision, recall, and F1 of identifying unanswerable questions in the dev set with various baselines that use different combinations of the question, original passage, and predicted context.

differences are the result of a few factors. The first is the fact that the more links required to answer a question, the more chances there are for failure to retrieve the necessary information. This is exacerbated by the pipeline nature of our baseline model. However, the spike in performance for questions requiring four or more links is caused by the number of unanswerable questions. Nearly half of the questions in that category are unanswerable, and the model largely predicts *No Answer* on those questions. Finally, the distribution of question types is different conditioned on the number of links. Questions that require more links often also require some form of discrete reasoning, which is more difficult for the model to handle.

Analyzing Different Answer Types Table 3.5 shows the performance broken down according to the type of answer each question has. The model performs worst on questions with numeric answers. This is due to the fact that these questions often require the model to do arithmetic to solve, which, as discussed above, the model struggles with relative to other types of questions.

Unanswerable Questions Table 3.6 shows how well a simple model can identify unanswerable questions with varying amounts of information. We set this up as a binary prediction, either answerable or not, and use a linear classifier that takes the BERT `CLS` token as input. We also include the result of always predicting unanswerable as a baseline. When the model can only see the question, it improves over the baseline by around 10 F1, meaning that there is some signal in the question alone, without any context.

Some types of questions are more likely to be unanswerable, such as those asking for information with regards to a specific year, i.e. *What was the population of New York in 1989?*. This is caused by Wikipedia more generally including current statistics, but not including a specific information for all previous years. Additionally adding the original passage does not significantly improve performance. This is not surprising, as the original passage always contains information relevant to the question, and the question annotators could see that text when writing the question.

3.4.4 Error Analysis

In order to better understand the challenges of the dataset, we manually analyzed 100 erroneous predictions by the model.

Incorrect context (39%) These are the cases where the model identified the correct links but selected the wrong portion of the linked document. It often selects semantically similar context but misses the crucial information, e.g. selecting the duration instead of end date.

Modeling errors (32%) These are the cases in which the context passed to the final QA model contained all of the necessary information, but the model failed to predict the correct answer. This occurred most commonly for questions requiring math, with the model including unnecessary dates in the computation, resulting in predictions that were orders of magnitude off. For example, predicting *-1984* when the question was asking for the age of a person.

Identifying unanswerable questions (24%) In these cases, the QA model was provided with related context that was missing crucial information, similar to the first class of errors. However, in this case, the full articles also did not contain the necessary information. In these cases the model often selected a related entity, ie for a question asking *In which ocean is the island nation located?*, the model predicted the island nation, *Papua New Guinea* as opposed to the ocean, which was not mentioned.

Insufficient Links (5%) These are cases where insufficient links were selected from the original passage, thus not providing enough information to answer the question. While the model can handle over-selection

Training Data	Links			QA	
	P	R	F1	EM	F1
IIRC	88	98	93	32.0	35.6
IIRC + S + D	85	79	82	24.6	28.0

Table 3.7: Results for link identification and QA when training the baseline model on IIRC and sampled questions from SQuAD (S) and DROP (D).

of links, we found that the vast majority of the time, the system correctly identified both the necessary and sufficient links, rarely over-predicting the required links.

3.5 Combined Evaluation

By construction, all the questions in IIRC require more than the original paragraph to answer. This means that a reading comprehension model built for IIRC does not actually have to detect *whether* more information is required than what is in the given paragraph, as it can always assume that this is true. In order to combat this bias, we recommend an additional, more stringent evaluation that combines IIRC with other reading comprehension datasets that do not require retrieving additional information. This is in line with recently-recommended evaluation methodologies for reading comprehension models [Talmor and Berant, 2019; Dua et al., 2019a].

In this section, we present the results of one such evaluation. Noting that IIRC has similar properties to both SQuAD 2.0 [Rajpurkar et al., 2018] and DROP [Dua et al., 2019b], and even similar question language in places, we sample questions from these datasets to form a combined dataset for training and evaluating our baseline model.

Sampling from SQuAD 2.0 and DROP To construct the data for the combined evaluation, we sample an additional 3360 questions from SQuAD 2.0 and DROP, so that they make up 20% of the questions in the new data. We sample from SQuAD 2.0 and DROP with a ratio of $3:1$ in order to match the distribution of numeric questions in IIRC and used a Wikifier [Cheng and Roth, 2013] to identify the links to Wikipedia articles in them.

Results We train the full baseline on IIRC augmented with sampled DROP and SQuAD data, and evaluate it on the IIRC dev set without any additional sampled data. We don't include any sampled data in the evaluation in order to make a direct comparison to IIRC to see how adding questions that don't require external context affects the model's ability to identify necessary context. We also include the results of running just the link identification model trained under each setting. We show the results in table 3.7. Adding the extra dimension of determining whether extra information is necessary causes the model to become less confident, significantly hurting recall on link selection. These missed predictions then propagate down the pipeline, resulting in a loss of almost 8% F_1 when compared to a model trained on just IIRC.

We also evaluated the combination model on a dev set with sampled SQuAD and DROP data to see how well the model learned to identify that no external information was necessary. Given that none of the SQuAD or DROP data requires external links, this evaluation could only negatively impact precision. We find that precision dropped by 8 points, compared to a drop of 28 points when the model trained only on IIRC was used, indicating that the model is able to learn to identify when no external information is required.

3.6 Summary

We introduced IIRC, a new dataset of incomplete-information reading comprehension questions. These questions require identifying what information is missing from a paragraph in order to answer a question, predicting where to find it, then synthesizing the retrieved information in complex ways. Our baseline model, built on top of state-of-the-art models for the most closely related existing datasets, performs quite poorly in this setting, even when given oracle retrieval results, and especially when combined with other reading comprehension datasets. IIRC both provides a promising new avenue for studying complex reading and retrieval problems and demonstrates that much more research is needed in this area.

Wilhelm Müller was born on 7 October 1794 at **Dessau**, the son of a tailor. In 1813-1814 he took part, as a volunteer in the Prussian army, in the national rising against **Napoleon**. He participated in the battles of **Lützen**, **Bautzen**, **Hanau** and **Kulm**. In 1814 he returned to his studies at Berlin. Müller's son, **Friedrich Max Müller**, was an English orientalist who founded the comparative study of religions.

Which battle Wilhelm Müller fought in while in the Prussian army had the most casualties?

Battle of Lützen (1813)
Napoleon lost 19,655 men, while the Prussians lost 8,500 men and the Russians lost 3,500 men

Battle of Bautzen
Losses on both sides totaled around 20,000.

Battle of Hanau
Overall, 4,500 French soldiers and 9,000 allied soldiers were lost in the battle.

Battle of Kulm
The French lost more than half of the pursuing force of 34,000; The allies lost approximately 13,000 soldiers.

Figure 3.1: An example from IIRC. At the top is a context paragraph which provides only partial information required to answer the question. The bold spans in the context indicate links to other Wikipedia pages. The colored boxes below the question show snippets from four of these pages that provide the missing information for answering the question. The answer is the underlined span.

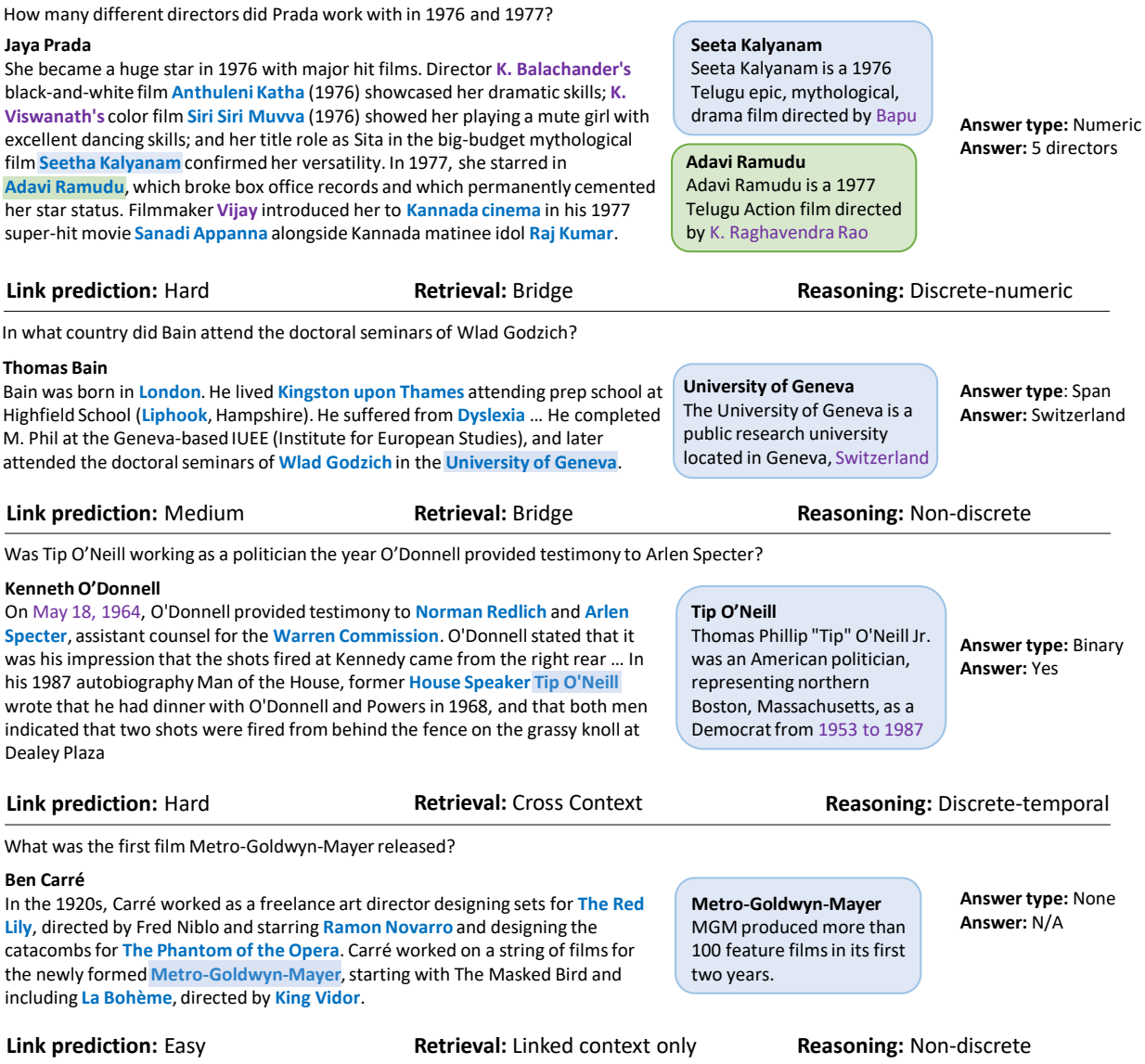


Figure 3.2: Examples from IIRC, labeled with what kinds of processing are required to answer each question. See Table 3.1 for more details. The passages on the left are the original passage, with bold spans indicating links. The highlighted sections contain the necessary context found in linked articles. Purple highlights indicate either the answer, for the second question, or the information used to compute the answer.

Chapter 4

Retrieval Augmentation Based on Downstream Performance

4.1 Introduction

Answering questions over a large text corpus typically requires retrieving information relevant to the question from the corpus, which is then used by a Question Answering (QA) model to arrive at the answer. Recent work [Guu et al., 2020b; Lewis et al., 2020b; Ni et al., 2020] relies on retrieval models that learn dense representations of questions and retrieval candidates [Karpukhin et al., 2020; Khattab and Zaharia, 2020] trained separately or jointly with the QA model. These learned retrieval models are more effective than those that use simple word overlap signals [Robertson and Zaragoza, 2009b; Chen et al., 2017], but they require the positive retrieval targets for each question labeled. It is often difficult, if not impossible, to exhaustively label all the facts relevant to answering a question in a large corpus of text. Consequently, even when the datasets provide retrieval labels, it is often the case that there exist alternative paths to the answer that are not labeled [Jhamtani and Clark, 2020], an example of which is shown in Figure 4.1. The common heuristic of considering all contexts that contain mentions of the answer span [Clark and Gardner, 2018; Lee et al., 2019] does not work when the QA task is not extractive (e.g.: when the answers are binary or require some numerical computation).

We propose to address this issue by augmenting the set of labeled retrieval targets with additional

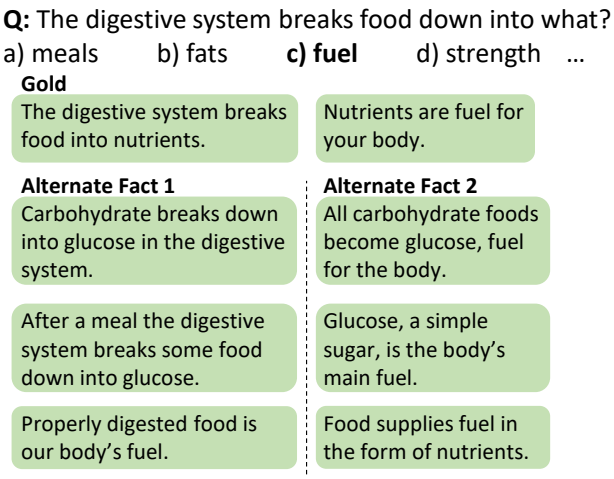


Figure 4.1: Retrieval annotations (gold) are often incomplete, only providing one of many relevant contexts. Alternative contexts can provide different views of the same information, providing more robust training data.

candidates that are not labeled as positive, but still provide sufficient information to answer the corresponding questions. Given question-answer pairs, and a QA model trained to maximize the likelihood of the correct answers conditioned on the labeled retrieval targets and the questions, we search for alternative contexts that also make the correct answers likely. Concretely, our search process finds those contexts not labeled as gold, that minimize the loss of the QA model. We consider these contexts as alternative retrieval targets, and train the retrieval model with the combination of these alternative contexts and the gold labeled contexts as positives. Our method is particularly effective for non-extractive QA tasks since it does not rely on answer-span overlaps.

We evaluate our approach on two multi-hop QA tasks, IIRC [Ferguson et al., 2020] and QASC [Khot et al., 2019a], and show that our search for relevant contexts guided by the performance of the QA model correctly identifies a relevant context 91% of the time on IIRC and 84% of the time on QASC (Table 4.2a). Augmenting the retrieval training data with the results from our search process increases recall on unseen questions, leading to an improvement in the downstream QA performance by 0.5 F_1 points on IIRC and 2.1 accuracy points on QASC (Section 4.3.2).

4.2 Method

Overview and Problem Our approach uses the standard two-step pipeline for open-domain QA seen in prior work. We first run a retrieval model that takes as input a question, q, and a large corpus of passages, C, and outputs a small subset of those passages, $c \subset C$, that contains sufficient information to answer the question. This subset is then passed to the second step: the QA model. This model takes as input the same question, q, and subset of passages, c, from the first step, and outputs an answer, a. Depending on the data, this answer can take many forms, such as a span from the context, a number, yes/no, or none of these if the question is unanswerable.

For each question, there may be many valid sets of context passages, where each set[1] contains all the information necessary to answer the question. We refer to individual sets as c_i^*, and the superset of all such sets as $c^* = \{c_1^* \dots c_n^*\}$. As seen in Figure 4.1, these different context sets may express different reasoning paths reaching the answer, or they may contain different ways of expressing the same reasoning path. However, most datasets just contain annotations of one such set per question, c_i^*. Our goal is to use these annotations to identify alternate, unannotated, relevant context, $\bar{c} \in c^* \setminus \{c_i^*\}$, for each question. These additional contexts is used to augment the retrieval training data.

Approach The goal of the retrieval model is to identify context that maximizes the probability of the correct answer when given to the QA model. When supervised data, c_i^*, is available, this is achieved by training the retrieval model to predict the input that the QA model is trained on i.e., $\theta_r = \arg\max_\theta P(c_i^*|q, \theta)$, and $\theta_q = \arg\max_\theta P(a|q, c_i^*, \theta)$, where the retriever and the QA models are parameterized by θ_r and θ_q. We refer to this initial QA model as the *base* QA model. When supervised data is not available, we can identify the retrieved contexts $\hat{c}$, by searching over the corpus for the contexts that maximize the probability of the correct answer under the base QA model:

$$\hat{c} = \arg\max_{c \subset C} P(a|q, c, \theta_q) \tag{4.1}$$

[1] We apply our approach to datasets containing questions that require multiple facts to answer, so we label *sets* of facts.

Based on this, for each question, we search over the corpus for the top k contexts, $\hat{c}_1 \ldots \hat{c}_k$, and add them as additional data augmentation when training a new retrieval model:

$$\hat{\theta_r} = \arg\max_{\theta} P(c_i^*|q,\theta) + \sum_{j=1}^{k} P(\hat{c}_j|q,\theta) \tag{4.2}$$

Lastly, we train a final QA model using the gold context, including the results of this new retrieval model to incorporate the updated training and make it more robust to noise:

$$\begin{aligned} c_r &= \arg\max_{c \in C} P(c|q,\hat{\theta_r}) \\ \hat{\theta_q} &= \arg\max_{\theta} P(a|q,\{c_i^*, c_r\},\theta) \end{aligned} \tag{4.3}$$

Labeling sets of facts Because we apply our approach to datasets containing questions that require multiple facts to answer, we need to label *sets* of facts, not individual ones. For this reason, we train our base QA models conditioned on sets of facts, and while both labeling new contexts with the base QA model, and retrieving contexts, we use beam search to output sets of facts. In order to prevent the base QA model from memorizing the gold contexts, we use a 10-fold cross-labeling approach.[2]

4.3 Experiments

We show the effect of our approach on two multi-hop QA datasets: IIRC [Ferguson et al., 2020] and QASC [Khot et al., 2019a].

4.3.1 Datasets and Setup

IIRC is a multi-hop QA open QA dataset, consisting of a mix of yes/no questions, span selection questions, unanswerable questions, and questions requiring discrete reasoning such as arithmetic or counting. Each question is associated with a paragraph, and requires both information from that paragraph, as well as information from one or more pages linked to from within that paragraph.

[2]We train ten models, each on 90% of the data, and use them to label the remaining 10%.

QASC is a multiple-choice, multi-hop QA dataset constructed from a corpus of 17M facts. Each question is written by composing two facts from the corpus, and includes eight answer choices.

eQASC [Jhamtani and Clark, 2020] includes a more exhaustive annotation of relevant contexts for QASC questions and enables a more accurate evaluation of retrieval performance on QASC.

Evaluation We report recall@10 and the final QA performance results that provide a more reliable evaluation of the retrieval performance. For eQASC, we use mean-average precision (MAP) of the positive examples.

Implementation Details Following prior work on IIRC [Ni et al., 2020], we adopt a pipeline approach consisting of three steps: link selection using RoBERTa-base, retrieval, and answer selection using NumNet++ [Ran et al., 2019]. For QASC, we initially filter the corpus using the two-step BM25 described in Khot et al. [2019a], selecting the top 1000 pairs of facts per answer choice. Similar to IIRC, we then select the top 10 pairs using a RoBERTa-base bi-encoder. Final QA model separately scores each answer choice using another RoBERTa-base model, and computes a softmax to get the final distribution over the choices.

4.3.2 Comparisons and Results

We compare our approach of identifying additional relevant context using QA loss with other retrieval baselines and alternate augmentation methods.

BM25: We use the top results from BM25 in lieu of training a supervised model with the annotated data. This is a commonly used heuristic when no retrieval annotations are available.

Sup$_{\mathbf{A}}$ Models are trained using just the annotated training data with no additional data provided.

Sup$_{\mathbf{A+BM25}}$ We augment the annotated training data with the top results from querying the corpus using BM25 with the question and answer.

Sup$_{\mathbf{A+R}}$ We augment the annotated training data with the top retrieval results conditioned on the question and correct answer. As in the QA-loss labeling approach, we use a 10-fold labeling procedure to prevent memorizing the annotated context.

Approach	QASC		IIRC		eQASC
	R@10	Acc	R@10	F1	MAP
BM25	45.1	71.9	18.0	42.0	36.0
Sup_{A}	46.1	71.8	39.5	51.1	41.9
$\text{Sup}_{\text{A + BM25}}$	41.7	69.3	38.0	49.2	40.3
$\text{Sup}_{\text{A + R}}$	46.2	71.5	39.3	51.0	35.4
$\text{Sup}_{\text{A + QA}}$	**47.8**	**73.9**	**40.3**	**51.6**	**43.7**
Prior Work	-	71.9	-	50.6	-

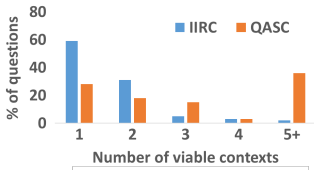
Table 4.1: Comparison of different retrieval models. R@10 and MAP are direct evaluations of retrieval performance, Acc is the performance of the final QA model trained given retrieval results. For IIRC, prior work is the state-of-the-art model [Ni et al., 2020] that uses the same QA model as our work. For QASC, prior work is RoBERTa-base model that uses the same model size as ours and is trained and evaluated on the same data used by Khashabi et al. [2020].

Approach	IIRC	QASC
BM25	38	41
Retrieval	39	45
QA Loss	91	84

(a)

Question type	QA	Span
Binary	100	-
Numeric	78	-
Span Selection	97	77
Span Compare	50	-

(b)



(c)

Table 4.2: (a) Manual analysis Accuracy of different approaches based on manual analysis on 100 examples for different context labeling approaches, (b) comparing span-selection retrieval baseline with our approach for different question types, and (c) Comparison of the number of relevant contexts in each dataset.

Main Results Table 4.1 compares our approach, $\text{Sup}_{\text{A+QA}}$, with the baselines and prior work.[3] Our approach results in improved performance on both datasets with a larger improvement on QASC over the baseline compared to IIRC. This is likely due to the fact that QASC has a much larger number of alternate contexts per question compared to IIRC (discussed below in oracle analysis). We generally see a correlation between retrieval recall of the gold annotations, performance on eQASC, and downstream accuracy, indicating that providing more accurate context to the downstream model does help with QA performance.

We manually labeled the accuracy of the top result for 100 questions for each approach (results in table 4.2a). We can see that using the QA model to label data significantly outperforms the other two approaches. In table 4.2b we also further break down the accuracy based on the different types of questions in IIRC. Our approach works well on *Binary* and *Numeric* questions, where the span heuristic cannot be applied. Our

[3]The state-of-the-art model [Khashabi et al., 2020] for QASC uses roughly 100x more parameters than us (with the results 89.6), but the same model with a comparable size as ours is significantly worse, 50.8. Therefore, we use the best-performing model that has the same size as ours.

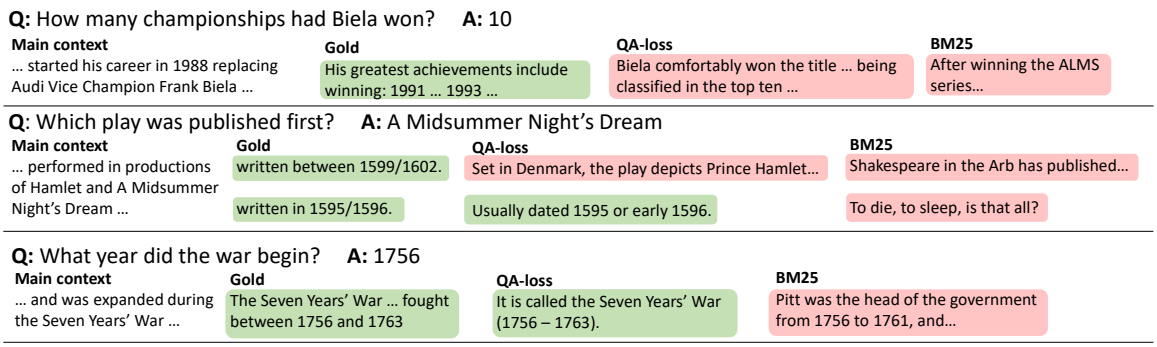


Figure 4.2: Example errors of our approach in IIRC. Relevant context is highlighted in green, and irrelevant context is in red.

approach also outperforms the it on *Span Selection* questions, where the answer is a span from the context. Although the heuristic can be applied on these questions, it often returns false positives. Our approach struggles with *Span Compare* questions, as discussed in more detail in Error Analysis below.

Oracle Analysis Figure 4.2c shows an oracle study of the same 100 questions from the previous section to determine how many alternate contexts were available in each dataset. For IIRC, we considered all sentences from the gold articles, and for QASC we considered the top twenty sentences according to BM25. QASC has a much higher ceiling for this form of data augmentation, as can be seen by the fact that 70% of questions have multiple relevant contexts, compared to IIRC where many questions have only a single context. Additionally, many of the questions in IIRC with exactly 2 contexts share a similar structure, seen in the third example in Figure 4.2. Although our approach is often able to identify this alternate context, using it to augment the data does not add much new information.

Error Analysis Figure 4.2 shows examples of problems our approach encounters in IIRC. The first question requires the model to count occurrences of an event, but the QA model instead selects context containing a textual expression of the answer. The second question is a *span compare* example. The model has to identify context containing attributes of two entities mentioned in the original paragraph, but takes a shortcut and and only selects context for the correct answer.

4.4 Summary

This chapter shows that using the loss of a QA model trained on a partial set of labeled contexts to search for alternative contexts for retrieval is an effective method for augmenting the retriever's training data. Our results present a more label-efficient training scheme for building supervised retrievers for QA. They also suggest that creators of datasets for open QA tasks that require supervised retrievers can better allocate their annotation budgets by obtaining retrieval labels for a small set of questions while maximizing the number of question-answer annotations.

Chapter 5

Question-Answer Pairs as Context

5.1 Introduction

Answering questions over a large text corpus typically requires retrieving information relevant to the question from the corpus, which is then used by a Question Answering model to arrive at the answer. Recent work [Guu et al., 2020b; Lewis et al., 2020b; Ni et al., 2020] relies on retrieval models that learn dense representations of questions and retrieval candidates [Karpukhin et al., 2020; Khattab and Zaharia, 2020] trained separately or jointly with the QA model. These learned retrieval models are more effective than those that use simple word overlap signals [Robertson and Zaragoza, 2009b; Chen et al., 2017]. In order to function effectively, they require the text to be broken down into manageable chunks. This is typically done by either splitting documents in the corpus into sentences, or into fixed size windows. However, as shown in Figure 5.1, often the necessary information that we want to retrieve is expressed in much smaller pieces. Thus, these methods of segmenting are inexact, and result in retrieval targets that have additional information causing noise for the retrieval model. Another direction is to use alternate sources of context, such as retrieving from a knowledge base [Verga et al., 2021; Févry et al., 2020], but that is limited by the scope of the KB. More recently, others such as Xiao et al. [2020]; Lewis et al. [2021a,b] have proposed directly retrieving from generated question-answer (QA) pairs and then using the retrieved answer as the answer to the question. QA pairs potentially provide fine-grained information, and thus do not have the limitation of fixed-length retrieval targets. Additionally, they are free-form, and thus not limited by a fixed schema, like KB tuples. However,

Context:

Fresno (/ˈfrɛznoʊ/ FREZ-noh), the county seat of Fresno County, is a city in the U.S. state of California. As of 2020, the city's population was 542,107, making it the fifth-largest city in California. It is approximately 170 miles (270 km) south of the state capital, Sacramento. The name Fresno means "ash tree" in Spanish, and an ash leaf is featured on the city's flag.

Question:

Is Fresno more populous than Sacramento?

Figure 5.1: Example question and context. The relevant information for the question, highlighted, makes up only a small portion of the context window typically used for retrieval.

this approach falls short on more complex questions such as those requiring multi-hop or discrete reasoning. Prior work has also explored using QA pairs as semantic representations of text [He et al., 2015; Michael et al., 2018; He et al., 2020], which has shown to be useful for a variety of tasks.

In this vein, we introduce QAPaC, Question-Answer Pairs As Context, a new approach to retrieving context in which documents are segmented into QA pairs, and then retrieval is done over a collection of QA pairs. Rather than directly using the resulting answer, we pass the retrieved context through a reader model trained for the end QA task in order to answer more complex questions. We evaluate this approach on IIRC, and show that it results in a 1.8 F1 point improvement compared to segmenting documents into either sentences or windows. Additionally, we show that using a combination of QA-pairs and sentences as inputs results in a further improvement of 1.3 F1 points.

5.2 Method

5.2.1 Overview and Problem

Our approach builds on top of the standard two-step pipeline for open-domain QA seen in prior work; first running a retrieval model to identify relevant context, and then passing that context into a QA model to return an answer.

The retrieval step takes as input a question, Q, and a set of documents, $D = \{d^0, d^1, \ldots d^k\}$, and outputs a small portion of those documents, $c \subset D$, that contains sufficient information to answer the question. In prior work, this is achieved by segmenting each document, d^i, into either sentences or fixed-size segments, $d^i = [d^i_0, d^i_1, \ldots d^i_n]$. The retrieval model then selects a subset of these segments from among all documents,

e.g. $c = \{d_0^0, d_4^1\}$.

For our experiments, we train a retrieval model that selects a set of k segments conditioned on the question. We simplify the problem by independently scoring each context.

$$\begin{aligned} c &= \underset{\binom{D_q}{k}}{\arg\max}\, P(d_0, \ldots d_k|Q) \\ &\approx \underset{\binom{D_q}{k}}{\arg\max}\, P(d_0|Q) \ldots P(d_k|Q) \end{aligned} \tag{5.1}$$

The retrieved subset, c, is then passed to the QA model for the second step. This model takes as input the question, Q, and subset of context, c, from the previous step, and outputs an answer, a. Depending on the data, this answer can take many forms, such as a span from the context, a number, yes/no, or none of these if the question is unanswerable.

For our experiments, we have gold annotated context spans in addition to the QA pairs, and as such, can directly train the retrieval model on gold segments. We then use the outputs of the retrieval model as input for the QA model at both train and test time.

5.2.2 Method

Our method specifically focuses on the document segmentation portion of the above approach. As shown in figure 5.1, It is difficult to select exactly the necessary information segmenting documents using fixed-size windows or sentences. These approaches often result in segments that contain either additional noisy information, or not enough information. We propose instead generating a set of simple, factual QA pairs for each document such that each QA pair targets a specific piece of information, $d^i = [q_0^i, q_1^i, \ldots q_m^i]$. We then use these QA-pairs, $D_q = \{q_0^0, \ldots q_m^0, q_0^1, q_1^1 \ldots q_m^k\}$, as retrieval candidates instead, e.g. $c = \{q_0^0, q_6^1\}$.

While we have gold annotated context spans, we do not have gold annotations for these generated questions, as there are often multiple questions generated per span. In order to train a retrieval model, we follow the approach of Ferguson et al. [2020], described in the previous chapter, to select training examples.

5.2.3 Question Generation

We first generate a set of QA pairs for each document in D. We follow the procedure use by Lewis et al. [2021b], and generate QA pairs by first selecting answer spans, then generating questions conditioned on those spans and their surrounding context.

Answer Selection Similar to Lewis et al. [2021b], we try three different approaches for selecting answers: using NER extractions, using a learned model, and extracting numeric tokens, and find that a combination of all three works best. Here we describe the architecture of the learned model. Given a tokenized document, $d = t_0, t_1, \ldots t_n$, we first encode the document using a language model, $LM(d) = e_0, e_1, \ldots e_n$. For a span, s_{ij}, starting at token i and ending at token j, we concatenate the embeddings of the first token, e_i and the last token, e_j. We then pass the resulting vector through a scoring layer, and compute a label for the span, l_{ij}, based on whether it scores above some threshold h,

$$l_{ij} = F([e_i, e_j]) > h \tag{5.2}$$

We consider all possible spans of up to length k for each document. This model is trained on the outputs of the NER system, and results in slightly better generalization than NER alone. Due to the nature of IIRC requiring discrete reasoning, we often want to include numeric information, so we augment the results of the other two systems with all tokens containing at least one numeric character. We present some analysis of these approaches in table 5.3.

Question Generation Once we've identified answer spans, we then generate questions conditioned on the span and surrounding context. For each answer span, $a = t_{\text{start}}, \ldots t_{\text{end}}$, we condition on context in a window of size w around the span; $c = t_{\text{start}-w}, \ldots t_{\text{end}+w}$. We prepend the answer and the context, and pass it as input to our generation model: $q = G([a, c])$. We do this for each document in D, resulting in D_q.

5.3 Experiment Setup

We evaluate the effects of our approach on IIRC [Ferguson et al., 2020].

5.3.1 IIRC

IIRC is a multi-hop QA open QA dataset, consisting of a mix of yes/no questions, span selection questions, unanswerable questions, and questions requiring discrete reasoning such as arithmetic or counting. Each question is associated with a passage, and requires both information from that passage, as well as information from one or more pages linked to from within that passage. When running the retrieval model, we concatenate the question and the provided passage to use as input, as opposed to just using the question.

5.3.2 Setup

Question Generation For our answer selection approaches, we use the spaCy NER system [Honnibal et al., 2020], trained on OntoNotes [Hovy et al., 2006], to make NER predictions. For the learned model, we use a RoBERTa-base model, and consider all spans up to length 10.

To generate questions, we use a BART-base [Lewis et al., 2020a] model conditioned on the answer, a, and context, c. We select as context a window around the answer. We prepend the answer to the context as input to the model. The model is trained on a combination of Natural Questions, SQuAD, and TriviaQA.

Retrieval Given the nature of IIRC, We follow the approach of Ferguson et al. [2020], and first select relevant linked documents, based on the original passage, then retrieve specific contexts from those documents. In order to identify relevant links, we first encode the passage using a RoBERTa-base model. We then represent each link as the concatenation of the encodings for the tokens immediately adjacent to the link. These are passed into a binary classification layer to select which links to use.

Then, once we've identified a set of documents, we retrieve context using a bi-encoder approach with two RoBERTa-base models. We first encode the question concatenated with the original passage using one model, using the resulting CLS embedding as the representation. We encode each context candidate using the other model, and compute scores for each as the dot product between the CLS encodings. We keep the top ten contexts across all relevant documents for each question.

We do not have annotated QA-pairs to train this model, so we use the approach of Ferguson et al. [2022] to obtain training data for the retrieval model. Specifically, we first train a QA model with QA-pairs generated from the gold context spans. We then search over the space of retrieval candidates, sorting them according to

the downstream QA score, and then select the contexts for which the downstream model is most likely to predict the correct answer.

Question Answering As in prior work on IIRC, we use NumNet+ [Ran et al., 2019] for the final QA step. This takes as input the question and retrieved context, and outputs either a span from the input, a number, a binary value, or *none* indicating an unanswerable question.

5.3.3 Models

We compare our approach with other methods of representing context. In addition to the approach we've outlined above, *QAPaC*, we consider four other approaches:

Sentence Documents in the retrieval corpus are segmented along sentence boundaries.

Window Documents in the retrieval corpus are segmented using a sliding window of 100 tokens with stride 25.

Ensemble Combination of this work and sentence context. Uses QA-pairs when predicting numeric and boolean answers, and sentences for all other answer types. Discussed in more detail below, and in table 5.2.

RePAQ The approach used in Lewis et al. [2021b], in which documents are first segmented into generated questions. The model then retrieves a context QA-pair and uses the answer to that question as the answer to the original question. Because RePAQ was not designed to handle unanswerable questions, We additionally augment it with a classifier to predict them. We use a RoBERTa-base model to encode the question and context, then pass the CLS token through a classification layer. Any questions that are labeled as answerable are then answered using RePAQ.

5.3.4 Evaluation Metrics

We use two evaluation metrics to compare model performance: retrieval Recall@10 and a numeracy-focused (macro-averaged) F1 score, originally proposed in Dua et al. [2019b], which measures overlap between a bag-of-words representation of the gold and predicted answers.

Approach	R@10	F1
Sentence	66	50.6
Window	**78**	50.3
RePAQ	-	23.7
QAPaC	61	52.4
Ensemble	71	**53.7**

Table 5.1: Comparison of different context representations. See section 5.3.4 for more details on the metrics.

Approach	Span	Num	Binary	None	Total
Sentence	46.7	35.9	50.4	**73.3**	50.6
Window	**47.1**	36.1	49.0	71.7	50.3
RePAQ	22.4	0.4	0.0	52.1	23.7
QAPaC	45.0	**40.8**	**57.0**	72.0	52.4
Ensemble	46.7	40.5	56.2	72.5	**53.7**

Table 5.2: Comparison of different context representations, broken down by answer type.

Because we don't have gold context QA-pair annotations, in order to report Recall@10, we manually label 100 examples from the dev set. We specifically select 100 examples in which some subset of the generated questions contains the information necessary to answer the question. We do this in order to isolate the retrieval performance from the question generation performance, which we also analyze in section 5.4.3. We use this same subset of examples to compute Recall@10 for all approaches. For the *QAPaC* approach we check whether the manually annotated questions appear in the retrieved context. For the *Window* and *Sentence* approaches, we check whether the context contains the originally annotated gold spans. Finally, for the *Ensemble* approach, we check the context of whichever model is used to make the final answer prediction.

5.4 Results

5.4.1 Context Baselines

We first compare the different approaches in terms of their retrieval performance and downstream QA performance, and show the results of these experiments in table 5.1. Both *Sentence* and *Window* perform comparably in terms of F1, while QAPaC outperforms both. We can see in table 5.2 that this is largely due to improved performance on numeric and binary questions, despite being worse at identifying unanswerable questions. Because of this, we include an ensemble approach in which the QA-pair context is used when

Approach	F1
NER	49.7
NER + Learned	50.1
NER + Learned + Numeric	52.4

Table 5.3: Comparison of different answer selection approaches. We report the downstream F1 when retrieving questions generated with each answer selection approach.

computing scores for numeric and binary answers and sentence context is used for all other answer types, resulting in further performance improvement. Finally, *RePAQ* performs very poorly on this dataset. This is expected, as it is largely unable to handle discrete reasoning, multi-hop questions, or binary answers.

For the Recall@10, *Window* performs the best, but also has much larger contexts compared to the other approaches. Because of this, even when it contains the correct information, the additional noise makes it more difficult for the QA model to predict the correct answer. Similarly, in the other direction, *QAPaC* has lower Recall@10, but also a smaller context size, making it easier for the QA model when it does have the correct information. Finally, we can see that the using the ensemble improves upon the Recall@10 compared to both *Sentence* and *QAPaC*, further confirming the fact that it is largely helping on numeric and binary questions.

5.4.2 Answer Selection

Here we explore the effects of different approaches to selecting answer spans for question generation. Following Lewis et al. [2021b], we try using NER and a learned model. Additionally, because IIRC contains a large number of questions requiring discrete reasoning, we try generating questions for each numeric token as well.

As can be seen in table 5.3, adding additional questions from each of the above approaches strictly improves performance. Using the learned model in addition to the NER model results in only a modest improvement, likely because the learned model is trained on the output of the NER model. As a result, while it generalizes slightly, it is not adding much additional information. However, adding in questions for numeric tokens results in a larger improvement. This is because the NER model does not consistently label numbers, and many questions in IIRC require numeric information.

Approach	F1
Gold context sentences	60.9
Gold context windows	55.7
Gold context questions	61.6
Oracle questions	67.0

Table 5.4: Oracle studies for different context representations. *Gold context sentences* and *windows* use the sentence or window that contains the gold annotated span. *Gold context questions* uses all QA-pairs that have an answer in the gold context span. *Oracle questions* uses manually labeled gold QA-pairs.

5.4.3 Question Generation

We analyze the potential effectiveness of using generated QA-pairs as context by using oracle annotations in order to determine the ceiling for this approach. All approaches are evaluated on the same subset of 100 examples for which we manually labeled gold context QA-pairs.

Gold Context We first compare the performance of the downstream QA model when run with sentences or windows containing the gold context spans vs using QA-pairs generated from those sentences. From table 5.4, we can see that there is not a substantial difference between using plain text compared to using the questions generated from that text. This is not surprising, as both approaches are different ways of representing the same information contained in the gold sentence. Note that multiple QA-pairs are generated for each gold sentence, only some of which are actually relevant, so this is equivalent to oracle retrieval for the sentence model, versus noisy retrieval for the QA-pair model.

Meanwhile, using windows containing the gold span does considerably worse than either of the other models. This is because the average sentence length is much lower than 100 tokens, so the windows contain extra unnecessary information that throws the model off.

Oracle Questions Also in table 5.4, we compare the downstream performance of using manually selected questions compared to sentences containing the gold context spans. This provides us with a ceiling for how well this approach can do with perfect question generation and retrieval. We can see that performance improves considerably when using oracle questions. This is not surprising, as the oracle questions contain exactly the information needed to answer the question, while the gold sentence annotation may contain some additional unnecessary information.

Span	Numeric	Binary	Total
0.85	0.92	1.0	0.90

Table 5.5: Analysis of question generation quality broken down based on question type. Examples are labeled based on whether or not the generated QA-pairs contain sufficient information to answer the question. QA-pairs were generated using the full NER + Learned + Numeric answer selection.

Direct Evaluation Finally, in table 5.5, we manually label 100 random examples based on whether or not any of the QA-pairs generated from the gold context contain the information necessary to answer the original question. We further break this down based on the type of question. The QA-pairs generally capture the necessary information. This is especially true for binary questions. We found that binary questions are tend to be more straightforward, factual question compared to the other types. They also comprised only a small sample of the 100 examples that were annotated. Meanwhile, we found that the approach does a better job of capturing the information for numeric questions compared to span questions. This is due to the fact that often the necessary information for numeric questions is expressed as a number, which we explicitly generate questions for.

5.5 Summary

This chapter shows that segmenting documents into question answer pairs allows retrieval models to better identify relevant information for question answering. Additionally, this representation can be used alongside a plain text representation of the document to further improve performance.

Chapter 6

Conclusion

In this dissertation we discussed three projects exploring the impact of context on question answering:

1. We introduced IIRC, a new dataset of incomplete information reading comprehension questions. These questions require models to perform discrete reasoning over multiple contexts. The questions were written by annotators without full access to information, resulting in naturally unanswerable questions, as well as reducing the lexical overlap between questions and context surrounding the answers. We showed that although humans are able to answer these questions with high accuracy, models struggle, even when provided gold context.

2. We introduced a new method for retrieval data augmentation based on a downstream QA model. Ranking potential contexts according to the likelihood of the downstream model predicting the correct answer is able to identify relevant context for over 90% of questions in IIRC. Using the contexts identified in this way as additional training data results in improved performance on both IIRC and QASC.

3. We introduced QAPaC, Question-Answer Pairs as Context. In this, we showed that representing documents as collections of generated QA pairs reduces noise for a retrieval model, resulting in a 1.8 F1 improvement in the downstream QA performance on IIRC. We additionally show that it allows the model to capture information orthogonal to that captured when the document is split into sentences, and that ensembling the two approaches results in a further 1.3 F1 improvement over just using QA

pairs.

6.1 Future Work

We proposed QAPaC as a method for representing documents as a collection of generated QA pairs. While this results in improved performance, there are still a lot of unexplored avenues. One such avenue is the encoding of the returned context. We simply concatenate the QA pairs with separator tokens, and pass them to the downstream QA model, which encodes them using a language model. This presents a couple of ideas. The first is to make more explicit use of the structure. Currently we are relying on the model to learn the importance of the structure, such as the fact that each question and answer are related, but aren't necessarily connected to the other QA pairs. This differs from what the model sees during pretraining, where separators denote sentence boundaries.

In a similar vein, the second idea is to adjust the LM pretraining so that it more closely matches the context it receives at test time. There has been some recent work on incorporating generated QA pairs into pretraining [Jia et al., 2022], but only as an additional signal alongside the original LM objective. One possible method of handling this could be to generate QA pairs during pretraining, and then pretrain on a QA objective instead by masking out the answer to a question, and then attempting to answer it conditioned on the other generated QA pairs. This would be useful both for this task, as it would make the pretraining data more closely resemble the inputs at test time, but it could also be useful as a general pretraining for question answering models. Additionally, this serves a similar purpose to SpanBERT [Joshi et al., 2020], where it attempts to mask more complete pieces of information, instead of arbitrary tokens.

Finally, in the current approach, the question generator is kept static after it has been trained. While it is trained on a dataset to attempt to generate simple factual questions, it might be the case that those do not capture information in the best way, especially as the domains are not necessarily the same. One option for updating the model would be to overgenerate context questions, and then update the generator based on which questions are selected as relevant context.

Bibliography

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. MathQA: Towards interpretable math word problem solving with operation-based formalisms. 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)*, pages 2357–2367, Minneapolis, Minnesota. Association for Computational Linguistics.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D. Manning. 2014. Modeling biological processes for reading comprehension. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1499–1510, Doha, Qatar. Association for Computational Linguistics.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 615–620, Doha, Qatar. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *ArXiv*, abs/2005.14165.

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

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

Xiao Cheng and Dan Roth. 2013. Relational Inference for Wikification. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

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)*, pages 845–855, Melbourne, Australia. Association for Computational Linguistics.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. Tydi qa: A benchmark for information-seeking question answering in typologically diverse languages. *ArXiv*, abs/2003.05002.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457.

Pradeep Dasigi, Nelson F Liu, Ana Marasovic, Noah A Smith, and Matt Gardner. 2019. Quoref: A reading comprehension dataset with questions requiring coreferential reasoning. 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)*, pages 5927–5934.

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)*, pages 4171–4186.

Bhuwan Dhingra, Kathryn Mazaitis, and William W. Cohen. 2017. Quasar: Datasets for question answering by search and reading. *ArXiv*, abs/1707.03904.

Dheeru Dua, Ananth Gottumukkala, Alon Talmor, Sameer Singh, and Matt Gardner. 2019a. ORB: An open reading benchmark for comprehensive evaluation of machine reading comprehension. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 147–153, Hong Kong, China. Association for Computational Linguistics.

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

James Ferguson, Matt Gardner, Hannaneh Hajishirzi, Tushar Khot, and Pradeep Dasigi. 2020. Iirc: A dataset of incomplete information reading comprehension questions. In *ACL*.

James Ferguson, Hannaneh Hajishirzi, Pradeep Dasigi, and Tushar Khot. 2022. Retrieval data augmentation informed by downstream question answering performance. In *Proceedings of the Fifth Fact Extraction and VERification Workshop (FEVER)*, pages 1–5, Dublin, Ireland. Association for Computational Linguistics.

Thibault Févry, Livio Baldini Soares, Nicholas FitzGerald, Eunsol Choi, and Tom Kwiatkowski. 2020. Entities as experts: Sparse memory access with entity supervision. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4937–4951, Online. Association for Computational Linguistics.

Bert F. Green, Alice K. Wolf, Carol L. Chomsky, and Kenneth Laughery. 1961. Baseball: an automatic question-answerer. In *IRE-AIEE-ACM '61 (Western)*.

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

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020b. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR.

Sanda M. Harabagiu, Dan I. Moldovan, Marius Pasca, Rada Mihalcea, Mihai Surdeanu, Razvan C. Bunescu, Roxana Girju, Vasile Rus, and Paul Morarescu. 2000. Falcon: Boosting knowledge for answer engines. In *TREC*.

Hangfeng He, Qiang Ning, and Dan Roth. 2020. QuASE: Question-answer driven sentence encoding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8743–8758, Online. Association for Computational Linguistics.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language.

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

Lynette Hirschman, Marc Light, Eric Breck, and John D. Burger. 1999. Deep read: A reading comprehension system. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 325–332, College Park, Maryland, USA. Association for Computational Linguistics.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spacy: Industrial-strength natural language processing in python.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 57–60, New York City, USA. Association for Computational Linguistics.

Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. 2014. A neural network for factoid question answering over paragraphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 633–644, Doha, Qatar. Association for Computational Linguistics.

Gautier Izacard and Edouard Grave. 2021. Distilling knowledge from reader to retriever for question answering. In *ICLR*.

Harsh Jhamtani and Peter Clark. 2020. Learning to explain: Datasets and models for identifying valid reasoning chains in multihop question-answering. In *EMNLP*.

Robin Jia, Mike Lewis, and Luke Zettlemoyer. 2022. Question answering infused pre-training of general-purpose contextualized representations. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 711–728, Dublin, Ireland. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

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)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP*.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. Unifiedqa: Crossing format boundaries with a single qa system. In *EMNLP*.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *SIGIR*.

Tushar Khot, Peter Clark, Michael Guerquin, Petre Jansen, and Shish Sabharwal. 2019a. Qasc: A dataset for question answering via sentence composition. In *AAAI*.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2019b. What's missing: A knowledge gap guided approach for multi-hop question answering. In *EMNLP*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. 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*, 7:453–466.

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*, pages 6086–6096, Florence, Italy. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *ArXiv*, abs/2104.08691.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020b. Retrieval-augmented generation for knowledge-intensive nlp tasks. *ArXiv*, abs/2005.11401.

Patrick Lewis, Pontus Stenetorp, and Sebastian Riedel. 2021a. Question and answer test-train overlap in open-domain question answering datasets.

Patrick Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Küttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. 2021b. Paq: 65 million probably-asked questions and what you can do with them.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, abs/2101.00190.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Julian Michael, Gabriel Stanovsky, Luheng He, Ido Dagan, and Luke Zettlemoyer. 2018. Crowdsourcing question-answer meaning representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 560–568, New Orleans, Louisiana. Association for Computational Linguistics.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*.

Sewon Min, Danqi Chen, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019a. A discrete hard EM approach for weakly supervised question answering. In *EMNLP*.

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

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019c. Multi-hop reading comprehension through question decomposition and rescoring. *ArXiv*, abs/1906.02916.

Dan I. Moldovan, Sanda M. Harabagiu, Marius Pasca, Rada Mihalcea, Richard Goodrum, Roxana Girju, and Vasile Rus. 1999. Lasso: A tool for surfing the answer net. In *TREC*.

Hwee Tou Ng, Leong Hwee Teo, and Jennifer Lai Pheng Kwan. 2000. A machine learning approach to answering questions for reading comprehension tests. In *2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 124–132, Hong Kong, China. Association for Computational Linguistics.

Ansong Ni, Matt Gardner, and Pradeep Dasigi. 2020. Mitigating false-negative contexts in multi-document question answering with retrieval marginalization. In *arXiv preprint arXiv:2103.12235*.

Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David A. McAllester. 2016. Who did what: A large-scale person-centered cloze dataset. In *EMNLP*.

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)*, pages 784–789.

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*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Qiu Ran, Yankai Lin, Peng Li, Jie Zhou, and Zhiyuan Liu. 2019. NumNet: Machine reading comprehension with numerical reasoning. In *Proceedings of EMNLP*.

Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203, Seattle, Washington, USA. Association for Computational Linguistics.

Ellen Riloff and Michael Thelen. 2000. A rule-based question answering system for reading comprehension tests. In *ANLP-NAACL 2000 Workshop: Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding Systems*.

Stephen Robertson and Hugo Zaragoza. 2009a. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Stephen Robertson and Hugo Zaragoza. 2009b. The probabilistic relevance framework: Bm25 and beyond. In *Foundations and Trends in Information Retrieval*.

Amrita Saha, Rahul Aralikatte, Mitesh M. Khapra, and Karthik Sankaranarayanan. 2018. Duorc: Towards complex language understanding with paraphrased reading comprehension. In *ACL*.

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

Minjoon Seo, Jinhyuk Lee, Tom Kwiatkowski, Ankur P. Parikh, Ali Farhadi, and Hannaneh Hajishirzi. 2019. Real-time open-domain question answering with dense-sparse phrase index. In *ACL*.

Alon Talmor and Jonathan Berant. 2019. MultiQA: An empirical investigation of generalization and transfer in reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4911–4921, Florence, Italy. Association for Computational Linguistics.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. NewsQA: A Machine Comprehension Dataset. *ArXiv*, abs/1611.09830.

Pat Verga, Haitian Sun, Livio Baldini Soares, and William Cohen. 2021. Adaptable and interpretable neural MemoryOver symbolic knowledge. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3678–3691, Online. Association for Computational Linguistics.

Ellen M. Voorhees and Dawn M. Tice. 1999. The trec-8 question answering track evaluation. In *TREC*.

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

W.A. Woods. 1977. Lunar rocks in natural English: Explorations in natural language question answering. In A. Zampolli, editor, *Linguistic Structures Processing*, pages 521–569. North-Holland, Amsterdam.

Jinfeng Xiao, Lidan Wang, Franck Dernoncourt, Trung Bui, Tong Sun, and Jiawei Han. 2020. Open-domain question answering with pre-constructed question spaces.

Sohee Yang and Minjoon Seo. 2021. Is retriever merely an approximator of reader? In *arXiv preprint arXiv:2010.10999*.

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*, pages 2369–2380.

Weijiang Yu, Yingpeng Wen, Fudan Zheng, and Nong Xiao. 2021. Improving math word problems with pre-trained knowledge and hierarchical reasoning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3384–3394, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xingdi Yuan, Jie Fu, Marc-Alexandre Cote, Yi Tay, Christopher Pal, and Adam Trischler. 2020. Interactive machine comprehension with information seeking agents. In *ACL*.

Zhuosheng Zhang, Jun jie Yang, and Hai Zhao. 2020. Retrospective reader for machine reading comprehension. *ArXiv*, abs/2001.09694.

Chen Zhao, Chenyan Xiong, Jordan Boyd-Graber, and Hal Daumé III. 2021. Distantly-supervised evidence retrieval enables question answerwing without evidence annotation. In *EMNLP*.