Did Aristotle Use a Laptop? A Question Answering Benchmark with Implicit Reasoning Strategies

Mor Geva[1,2], Daniel Khashabi[2], Elad Segal[1], Tushar Khot[2], Dan Roth[3], Jonathan Berant[1,2]

[1]Tel Aviv University [2]Allen Institute for AI [3]University of Pennsylvania
morgeva@mail.tau.ac.il, {danielk,tushark}@allenai.org,
elad.segal@gmail.com, danroth@seas.upenn.edu
joberant@cs.tau.ac.il

Abstract

A key limitation in current datasets for *multi-hop reasoning* is that the required steps for answering the question are mentioned in it *explicitly*. In this work, we introduce StrategyQA, a question answering (QA) benchmark where the required reasoning steps are *implicit* in the question, and should be inferred using a *strategy*. A fundamental challenge in this setup is how to elicit such creative questions from crowdsourcing workers, while covering a broad range of potential strategies. We propose a data collection procedure that combines term-based priming to inspire annotators, careful control over the annotator population, and adversarial filtering for eliminating reasoning shortcuts. Moreover, we annotate each question with (1) a decomposition into reasoning steps for answering it, and (2) Wikipedia paragraphs that contain the answers to each step. Overall, StrategyQA includes 2,780 examples, each consisting of a strategy question, its decomposition, and evidence paragraphs. Analysis shows that questions in StrategyQA are short, topic-diverse, and cover a wide range of strategies. Empirically, we show that humans perform well (87%) on this task, while our best baseline reaches an accuracy of $\sim 66\%$.

1 Introduction

Developing models that successfully reason over multiple parts of their input has attracted substantial attention recently, leading to the creation of many multi-step reasoning Question Answering (QA) benchmarks (Welbl et al., 2018; Talmor and Berant, 2018; Khashabi et al., 2018; Yang et al., 2018; Dua et al., 2019; Suhr et al., 2019).

Commonly, the language of questions in such benchmarks *explicitly* describes the process for deriving the answer. For instance (Figure 1, Q2), the question *Was Aristotle alive when the laptop was invented?* explicitly specifies the required reasoning steps. However, in real-life questions, reasoning is often *implicit*. For example, the question *Did Aristotle use a laptop?* (Q1) can be answered using the same steps, but the model must infer the *strategy* for answering the question–*temporal* comparison, in this case.

Answering implicit questions poses several challenges compared to answering their explicit counterparts. First, retrieving the context is difficult as there is little overlap between the question and its context (Figure 1, Q1 and 'E'). Moreover, questions tend to be short, lowering the possibility of the model exploiting shortcuts in the language of the question. In this work, we introduce StrategyQA, a Boolean QA benchmark focusing on implicit multi-hop reasoning for *strategy questions*, where a *strategy* is the ability to infer from a question its atomic sub-questions. In contrast to previous benchmarks (Khot et al., 2020a; Yang et al., 2018), questions in StrategyQA are not limited to predefined decomposition patterns and cover a wide range of strategies that humans apply when answering questions.

Eliciting strategy questions using crowdsourcing is non-trivial. First, authoring such questions requires *creativity*. Past work often collected multi-hop questions by showing workers an entire context, which led to limited creativity and high lexical overlap between questions and contexts and consequently to reasoning shortcuts (Khot et al., 2020a; Yang et al., 2018). An alternative approach, applied in Natural Questions

Transactions of the Association for Computational Linguistics, vol. 9, pp. 346–361, 2021. https://doi.org/10.1162/tacl_a_00370
Action Editor: Kristina Toutanova. Submission batch: 10/2020; Revision batch: 12/2020; Published 4/2021.

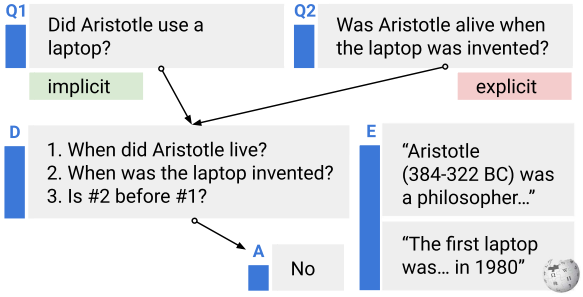

Figure 1: Questions in StrategyQA (Q1) require *implicit* decomposition into reasoning steps (D), for which we annotate supporting evidence from Wikipedia (E). This is in contrast to multi-step questions that *explicitly* specify the reasoning process (Q2).

(Kwiatkowski et al., 2019) and MS-MARCO (Nguyen et al., 2016), overcomes this by collecting real user questions. However, can we elicit creative questions independently of the context and without access to users?

Second, an important property in StrategyQA is that questions entail *diverse* strategies. While the example in Figure 1 necessitates temporal reasoning, there are many possible strategies for answering questions (Table 1). We want a benchmark that exposes a broad range of strategies. But crowdsourcing workers often use repetitive patterns, which may limit question diversity.

To overcome these difficulties, we use the following techniques in our pipeline for eliciting strategy questions: (a) we prime crowd workers with random Wikipedia terms that serve as a minimal context to inspire their imagination and increase their creativity; (b) we use a large set of annotators to increase question diversity, limiting the number of questions a single annotator can write; and (c) we continuously train adversarial models during data collection, slowly increasing the difficulty in question writing and preventing recurring patterns (Bartolo et al., 2020).

Beyond the questions, as part of StrategyQA, we annotated: (a) *question decompositions*: a sequence of steps sufficient for answering the question ('D' in Figure 1), and (b) *evidence* paragraphs: Wikipedia paragraphs that contain the answer to *each* decomposition step ('E' in Figure 1). StrategyQA is the first QA dataset to provide decompositions and evidence annotations for each individual step of the reasoning process.

Our analysis shows that StrategyQA necessitates reasoning on a wide variety of knowledge domains (physics, geography, etc.) and logical operations (e.g., number comparison). Moreover, experiments show that StrategyQA poses a combined challenge of retrieval and QA, and while humans perform well on these questions, even strong systems struggle to answer them.

In summary, the contributions of this work are:

1. Defining *strategy* questions—a class of question requiring *implicit* multi-step reasoning.

2. StrategyQA, the first benchmark for implicit multi-step QA, that covers a diverse set of reasoning skills. StrategyQA consists of 2,780 questions, annotated with their decomposition and per-step evidence.

3. A novel annotation pipeline designed to elicit quality strategy questions, with minimal context for priming workers.

The dataset and codebase are publicly available at `https://allenai.org/data/strategyqa`.

2 Strategy Questions

2.1 Desiderata

We define strategy questions by characterizing their desired properties. Some properties, such as whether the question is answerable, also depend on the context used for answering the question. In this work, we assume this context is a corpus of documents, specifically, Wikipedia, which we assume provides correct content.

Multi-step Strategy questions are multi-step questions, that is, they comprise a sequence of *single-step questions*. A single-step question is either (a) a question that can be answered from a short text fragment in the corpus (e.g., steps 1 and 2 in Figure 1), or (b) a logical operation over answers from previous steps (e.g., step 3 in Figure 1). A strategy question should have at least two steps for deriving the answer. Example multi- and single- step questions are provided in Table 2. We define the reasoning process structure in §2.2.

Feasible Questions should be answerable from paragraphs in the corpus. Specifically, for each

Question	Implicit facts
Can one spot helium? (No)	Helium is a gas, Helium is odorless, Helium is tasteless, Helium has no color.
Would Hades and Osiris hypothetically compete for real estate in the Underworld? (Yes)	Hades was the Greek god of death and the Underworld. Osiris was the Egyptian god of the Underworld.
Would a monocle be appropriate for a cyclop? (Yes)	Cyclops have one eye. A monocle helps one eye at a time.
Should a finished website have lorem ipsum paragraphs? (No)	Lorem Ipsum paragraphs are meant to be temporary. Web designers always remove lorem ipsum paragraphs before launch.
Is it normal to find parsley in multiple sections of the grocery store? (Yes)	Parsley is available in both fresh and dry forms. Fresh parsley must be kept cool. Dry parsley is a shelf stable product.

Table 1: Example strategy questions and the implicit facts needed for answering them.

Question	MS	IM	Explanation
Was Barack Obama born in the United States? (Yes)			The question explicitly states the required information for the answer–the birth place of Barack Obama. The answer is likely to be found in a single text fragment in Wikipedia.
Do cars use drinking water to power their engine? (No)			The question explicitly states the required information for the answer–the liquid used to power car engines. The answer is likely to be found in a single text fragment in Wikipedia.
Are sharks faster than crabs? (Yes)	✓		The question explicitly states the required reasoning steps: 1) How fast are sharks? 2) How fast are crabs? 3) Is #1 faster than #2?
Was Tom Cruise married to the female star of Inland Empire? (No)	✓		The question explicitly states the required reasoning steps: 1) Who is the female star of Inland Empire? 2) Was Tom Cruise married to #2?
Are more watermelons grown in Texas than in Antarctica? (Yes)	✓	✓	The answer can be derived through geographical/botanical reasoning that the climate in Antarctica does not support growth of watermelons.
Would someone with a nosebleed benefit from Coca? (Yes)	✓	✓	The answer can be derived through biological reasoning that Coca constricts blood vessels, and therefore, serves to stop bleeding.

Table 2: Example questions demonstrating the multi-step (MS) and implicit (IM) properties of strategy questions.

reasoning step in the sequence, there should be sufficient evidence from the corpus to answer the question. For example, the answer to the question *Would a monocle be appropriate for a cyclop?* can be derived from paragraphs stating that cyclops have one eye and that a monocle is used by one eye at the time. This information is found in our corpus, Wikipedia, and thus the question is feasible. In contrast, the question *Does Justin Beiber own a Zune?* is *not* feasible, because answering it requires going through Beiber's belongings, and this information is unlikely to be found in Wikipedia.

Implicit A key property distinguishing strategy questions from prior multi-hop questions is their implicit nature. In explicit questions, each step in the reasoning process can be inferred from the *language* of the question directly. For example, in Figure 1, the first two questions are explicitly stated, one in the main clause and one in the adverbial clause. Conversely,

reasoning steps in strategy questions require going beyond the language of the question. Due to language variability, a precise definition of implicit questions based on lexical overlap is elusive, but a good rule-of-thumb is the following: If the question decomposition can be written with a vocabulary limited to words from the questions, their inflections, and function words, then it is an explicit question. If new content words must be introduced to describe the reasoning process, the question is implicit. Examples for implicit and explicit questions are in Table 2.

Definite A type of questions we wish to avoid are *non-definitive* questions, such as *Are hamburgers considered a sandwich?* and *Does chocolate taste better than vanilla?* for which there is no clear answer. We would like to collect questions where the answer is definitive or, at least, very likely, based on the corpus. For example, consider the question *Does wood conduct electricity?*. Although it is possible that a damp wood will conduct electricity, the answer is generally *no*.

To summarize, strategy questions are multi-step questions with implicit reasoning (a strategy) and a definitive answer that can be reached given a corpus. We limit ourselves to Boolean yes/no questions, which limits the output space, but lets us focus on the complexity of the questions, which is the key contribution. Example strategy questions are in Table 1, and examples that demonstrate the mentioned properties are in Table 2. Next (§2.2), we describe additional structures annotated during data collection.

2.2 Decomposing Strategy Questions

Strategy questions involve complex reasoning that leads to a yes/no answer. To guide and evaluate the QA process, we annotate every example with a description of the expected reasoning process.

Prior work used *rationales* or *supporting facts*, namely, text snippets extracted from the context (DeYoung et al., 2020; Yang et al., 2018; Kwiatkowski et al., 2019; Khot et al., 2020a) as evidence for an answer. However, reasoning can rely on elements that are not explicitly expressed in the context. Moreover, answering a question based on relevant context does not imply that the model performs reasoning properly (Jiang and Bansal, 2019).

Question	Decomposition
Did the Battle of Peleliu or the Seven Days Battles last longer?	(1) How long did *the Battle of Peleliu last*? (2) How long did *the Seven Days Battle last*? (3) Which is *longer* of #1, #2?
Can the President of Mexico vote in New Mexico primaries?	(1) What is **the citizenship requirement** for *voting in New Mexico*? (2) What is **the citizenship requirement** of any *President of Mexico*? (3) Is #2 the same as #1?
Can a microwave melt a Toyota Prius battery?	(1) What **kind of** *battery* does a *Toyota Prius* use? (2) What **type of material** is #1 made out of? (3) What is the **melting point** of #2? (4) Can a *microwave's* **temperature** reach at least #3?
Would it be common to find a penguin in Miami?	(1) Where is a typical *penguin's* **natural habitat**? (2) What **conditions** make #1 suitable for *penguins*? (3) Are all of #2 present in *Miami*?

Table 3: Explicit (row 1) and strategy (rows 2–4) question decompositions. We mark words that are explicit (*italic*) or implicit in the input (**bold**).

Inspired by recent work (Wolfson et al., 2020), we associate every question-answer pair with a *strategy question decomposition*. A decomposition of a question q is a sequence of n steps $\langle s^{(1)}, s^{(2)}, \ldots, s^{(n)} \rangle$ required for computing the answer to q. Each step $s^{(i)}$ corresponds to a single-step question and may include special *references*, which are placeholders referring to the result of a previous step $s^{(j)}$. The last decomposition step (i.e., $s^{(n)}$) returns the final answer to the question. Table 3 shows decomposition examples.

Wolfson et al. (2020) targeted explicit multi-step questions (first row in Table 3), where the decomposition is restricted to a small vocabulary derived almost entirely from the original question. Conversely, decomposing strategy questions

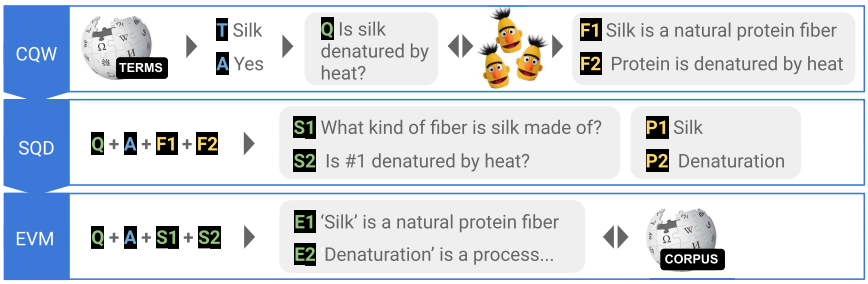

Figure 2: Overview of the data collection pipeline. First (CQW, §3.1), a worker is presented with a term (T) and an expected answer (A) and writes a question (Q) and the facts (F1, F2) required to answer it. Next, the question is decomposed (SQD, §3.2) into steps (S1, S2) along with Wikipedia page titles (P1, P2) that the worker expects to find the answer in. Last (EVM, §3.3), decomposition steps are matched with evidence from Wikipedia (E1, E2).

requires using implicit knowledge, and thus decompositions can include any token that is needed for describing the implicit reasoning (rows 2–4 in Table 3). This makes the decomposition task significantly harder for strategy questions.

In this work, we distinguish between two types of required actions for executing a step. *Retrieval*, a step that requires retrieval from the corpus, and *operation*, a logical function over answers to previous steps. In the second row of Table 3, the first two steps are retrieval steps, and the last step is an operation. A decomposition step can require both retrieval and an operation (see last row in Table 3).

To verify that steps are valid single-step questions that can be answered using the corpus (Wikipedia), we collect *supporting evidence* for each retrieval step and annotate operation steps. A supporting evidence is one or more paragraphs that provide an answer to the retrieval step.

In summary, each example in our dataset contains a) a strategy question, b) the strategy question decomposition, and c) supporting evidence per decomposition step. Collecting strategy questions and their annotations is the main challenge of this work, and we turn to this next.

3 Data Collection Pipeline

Our goal is to establish a procedure for collecting strategy questions and their annotations at scale. To this end, we build a multi-step crowdsourcing[1] pipeline designed for encouraging worker creativity, while preventing biases in the data.

[1]We use Amazon Mechanical Turk as our framework.

We break the data collection into three tasks: question writing (§3.1), question decomposition (§3.2), and evidence matching (§3.3). In addition, we implement mechanisms for quality assurance (§3.4). An overview of the data collection pipeline is in Figure 2.

3.1 Creative Question Writing (CQW)

Generating natural language annotations through crowdsourcing (e.g., question generation) is known to suffer from several shortcomings. First, when annotators generate many instances, they use recurring patterns that lead to biases in the data. (Gururangan et al., 2018; Geva et al., 2019). Second, when language is generated conditioned on a long context, such as a paragraph, annotators use similar language (Kwiatkowski et al., 2019), leading to high lexical overlap and hence, inadvertently, to an easier problem. Moreover, a unique property of our setup is that we wish to cover a *broad and diverse* set of strategies. Thus, we must discourage repeated use of the same strategy.

We tackle these challenges on multiple fronts. First, rather than using a long paragraph as context, we prime workers to write questions given single terms from Wikipedia, reducing the overlap with the context to a minimum. Second, to encourage diversity, we control the population of annotators, making sure a large number of annotators contribute to the dataset. Third, we use *model-in-the-loop* adversarial annotations (Dua et al., 2019; Khot et al., 2020a; Bartolo et al., 2020) to filter our questions, and only accept questions that fool our models. While some model-in-the-loop approaches use fixed pre-trained models to eliminate ‘‘easy’’ questions, we continuously

update the models during data collection to combat the use of repeated patterns or strategies.

We now provide a description of the task, and elaborate on these methods (Figure 2, upper row).

Task description Given a term (e.g., *silk*), a description of the term, and an expected answer (yes or no), the task is to write a strategy question about the term with the expected answer, and the facts required to answer the question.

Priming with Wikipedia Terms Writing strategy questions from scratch is difficult. To inspire worker creativity, we ask to write questions about terms they are familiar with or can easily understand. The terms are titles of ''popular''[2] Wikipedia pages. We provide workers only with a short description of the given term. Then, workers use their background knowledge and Web search skills to form a strategy question.

Controlling the Answer Distribution We ask workers to write questions where the answer is set to be 'yes' or 'no'. To balance the answer distribution, the expected answer is dynamically sampled inversely proportional to the ratio of 'yes' and 'no' questions collected until that point.

Model-in-the-Loop Filtering To ensure questions are challenging and reduce recurring language and reasoning patterns, questions are only accepted when verified by two sets of online *solvers*. We deploy a set of 5 pre-trained models (termed PTD) that check if the question is too easy. If at least 4 out of 5 answer the question correctly, it is rejected. Second, we use a set of 3 models (called FNTD) that are continuously fine-tuned on our collected data and are meant to detect biases in the current question set. A question is rejected if all 3 solvers answer it correctly. The solvers are ROBERTA (Liu et al., 2019) models fine-tuned on different auxiliary datasets; details in §5.1.

Auxiliary Sub-Task We ask workers to provide the facts required to answer the question they have written, for several reasons: 1) it helps workers frame the question writing task and describe the reasoning process they have in mind, 2) it helps reviewing their work, and 3) it provides useful information for the decomposition step (§3.2).

[2]We filter pages based on the number of contributors and the number of backward links from other pages.

3.2 Strategy Question Decomposition (SQD)

Once a question and the corresponding facts are written, we generate the strategy question decomposition (Figure 2, middle row). We annotate decompositions *before* matching evidence in order to avoid biases stemming from seeing the context.

The decomposition strategy for a question is not always obvious, which can lead to undesirable explicit decompositions. For example, a possible explicit decomposition for Q1 (Figure 1) might be (1) *What items did Aristotle use?* (2) *Is laptop in #1?*; but the first step is not feasible. To guide the decomposition, we provide workers with the facts written in the CQW task to show the strategy of the question author. Evidently, there can be many valid strategies and the same strategy can be phrased in multiple ways—the facts only serve as a soft guidance.

Task Description Given a strategy question, a yes/no answer, and a set of facts, the task is to write the steps needed to answer the question.

Auxiliary Sub-task We observe that in some cases, annotators write explicit decompositions, which often lead to infeasible steps that cannot be answered from the corpus. To help workers avoid explicit decompositions, we ask them to specify, for each decomposition step, a Wikipedia page they expect to find the answer in. This encourages workers to write decomposition steps for which it is possible to find answers in Wikipedia, and leads to feasible strategy decompositions, with only a small overhead (the workers are not required to read the proposed Wikipedia page).

3.3 Evidence Matching (EVM)

We now have a question and its decomposition. To ground them in context, we add a third task of evidence matching (Figure 2, bottom row).

Task Description Given a question and its decomposition (a list of single-step questions), the task is to find evidence paragraphs on Wikipedia for each retrieval step. Operation steps that do not require retrieval (§2.2) are marked as *operation*.

Controlling the Matched Context Workers search for evidence on Wikipedia. We index Wikipedia[3] and provide a search interface where workers can drag-and-drop paragraphs from the

[3]We use the Wikipedia Cirrus dump from 11/05/2020.

results shown on the search interface. This guarantees that annotators choose paragraphs we included in our index, at a pre-determined paragraph-level granularity.

3.4 Data Verification Mechanisms

Task Qualifications For each task, we hold qualifications that test understanding of the task, and manually review several examples. Workers who follow the requirements are granted access to our tasks. Our qualifications are open to workers from English-speaking countries who have high reputation scores. Additionally, the authors regularly review annotations to give feedback and prevent noisy annotations.

Real-time Automatic Checks For CQW, we use heuristics to check question validity, for example, whether it ends with a question mark, and that it doesn't use language that characterizes explicit multi-hop questions (for instance, having multiple verbs). For SQD, we check that the decomposition structure forms a directed acyclic graph, that is: (i) each decomposition step is referenced by (at least) one of the following steps, such that all steps are reachable from the last step; and (ii) steps don't form a cycle. In the EVM task, a warning message is shown when the worker marks an intermediate step as an operation (an unlikely scenario).

Inter-task Feedback At each step of the pipeline, we collect feedback about previous steps. To verify results from the CQW task, we ask workers to indicate whether the given answer is incorrect (in the SQD, EVM tasks), or if the question is not definitive (in the SQD task) (§2.1). Similarly, to identify non-feasible questions or decompositions, we ask workers to indicate if there is no evidence for a decomposition step (in the EVM task).

Evidence Verification Task After the EVM step, each example comprises a question, its answer, decomposition, and supporting evidence. To verify that a question can be answered by executing the decomposition steps against the matched evidence paragraphs, we construct an additional evidence verification task (EVV). In this task, workers are given a question, its decomposition and matched paragraphs, and are asked to answer the question in each decomposition step purely based on the provided paragraphs. Running EVV on a subset of examples during data collection helps identify issues in the pipeline and in worker performance.

4 The StrategyQA Dataset

We run our pipeline on 1,799 Wikipedia terms, allowing a maximum of 5 questions per term. We update our online fine-tuned solvers (Fntd) every 1K questions. Every question is decomposed once, and evidence is matched for each decomposition by 3 different workers. The cost of annotating a full example is $4.

To encourage diversity in strategies used in the questions, we recruited new workers throughout data collection. Moreover, periodic updates of the online solvers prevent workers from exploiting shortcuts, since the solvers adapt to the training distribution. Overall, there were 29 question writers, 19 decomposers, and 54 evidence matchers participating in the data collection.

We collected 2,835 questions, out of which 55 were marked as having an incorrect answer during SQD (§3.2). This results in a collection of 2,780 verified strategy questions, for which we create an annotator-based data split (Geva et al., 2019). We now describe the dataset statistics (§4.1), analyze the quality of the examples (§4.2), and explore the reasoning skills in StrategyQA (§4.3).

4.1 Dataset Statistics

We observe (Table 4) that the answer distribution is roughly balanced (yes/no). Moreover, questions are short (< 10 words), and the most common trigram occurs in roughly 1% of the examples. This indicates that the language of the questions is both simple and diverse. For comparison, the average question length in the multi-hop datasets HotpotQA (Yang et al., 2018) and ComplexWebQuestions (Talmor and Berant, 2018) is 13.7 words and 15.8 words, respectively. Likewise, the top trigram in these datasets occurs in 9.2% and 4.8% of their examples, respectively.

More than half of the generated questions are filtered by our solvers, pointing to the difficulty of generating good strategy questions. We release all 3,305 filtered questions as well.

To characterize the *reasoning complexity* required to answer questions in StrategyQA, we examine the decomposition length and the number of evidence paragraphs. Figure 3 and Table 4 (bottom) show the distributions of these properties

	Train	Test
# of questions	2290	490
% ‘‘yes’’ questions	46.8%	46.1%
# of unique terms	1333	442
# of unique decomposition steps	6050	1347
# of unique evidence paragraphs	9251	2136
# of occurrences of the top trigram	31	5
# of question writers	23	6
# of filtered questions	2821	484
Avg. question length (words)	9.6	9.8
Avg. decomposition length (steps)	2.93	2.92
Avg. # of paragraphs per question	2.33	2.29

Table 4: StrategyQA statistics. Filtered questions were rejected by the solvers (§3.1). The train and test sets of question writers are disjoint. The ‘‘top trigram’’ is the most common trigram.

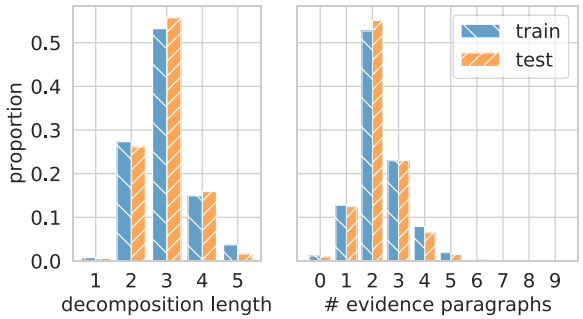

Figure 3: The distributions of decomposition length (left) and the number of evidence paragraphs (right). The majority of the questions in StrategyQA require a reasoning process comprised of ≥ 3 steps, of which about 2 steps involve retrieving external knowledge.

are centered around 3-step decompositions and 2 evidence paragraphs, but a considerable portion of the dataset requires more steps and paragraphs.

4.2 Data Quality

Do questions in StrategyQA require multi-step implicit reasoning? To assess the quality of questions, we sampled 100 random examples from the training set, and had two experts (authors) independently annotate whether the questions satisfy the desired properties of strategy questions (§2.1). We find that most of the examples (81%) are valid multi-step implicit questions, 82% of questions are implicit, and 95.5% are multi-step (Table 5).

	multi-step	single-step	
implicit	81	1	82
explicit	14.5	3.5	18
	95.5	4.5	100

Table 5: Distribution over the implicit and multi-step properties (§2) in a sample of 100 StrategyQA questions, annotated by two experts (we average the expert decisions). Most questions are *multi-step* and *implicit*. Annotator agreement is substantial for both the implicit ($\kappa = 0.73$) and multi-step ($\kappa = 0.65$) properties.

Do questions in StrategyQA have a definitive answer? We let experts review the answers to 100 random questions, allowing access to the Web. We then ask them to state for every question whether they agree or disagree with the provided answer. We find that the experts agree with the answer in 94% of the cases, and disagree only in 2%. For the remaining 4%, either the question was ambiguous, or the annotators could not find a definite answer on the Web. Overall, this suggests that questions in StrategyQA have clear answers.

What is the quality of the decompositions? We randomly sampled 100 decompositions and asked experts to judge their quality. Experts judged if the decomposition is explicit or utilizes a strategy. We find that 83% of the decompositions validly use a strategy to break down the question. The remaining 17% decompositions are explicit, however, in 14% of the cases the original question is already explicit. Second, experts checked if the phrasing of the decomposition is ‘‘natural’’, that is, it reflects the decomposition of a person that does not already know the answer. We find that 89% of the decompositions express a ‘‘natural’’ reasoning process, while 11% may depend on the answer. Last, we asked experts to indicate any potential logical flaws in the decomposition, but no such cases occurred in the sample.

Would different annotators use the same decomposition strategy? We sample 50 examples, and let two different workers decompose the questions. Comparing the decomposition pairs, we find that a) for all pairs,

the last step returns the same answer, b) in 44 out of 50 pairs, the decomposition pairs follow the same reasoning path, and c) in the other 6 pairs, the decompositions either follow a different reasoning process (5 pairs) or one of the decompositions is explicit (1 pair). This shows that different workers usually use the same strategy when decomposing questions.

Is the evidence for strategy questions in Wikipedia? Another important property is whether questions in StrategyQA can be answered based on context from our corpus, Wikipedia, given that questions are written independently of the context. To measure evidence coverage, in the EVM task (§3.3), we provide workers with a checkbox for every decomposition step, indicating whether only partial or no evidence could be found for that step. Recall that three different workers match evidence for each decomposition step. We find that 88.3% of the questions are fully covered: Evidence was matched for each step by some worker. Moreover, in 86.9% of the questions, at least one worker found evidence for *all* steps. Last, in only 0.5% of the examples were all three annotators unable to match evidence for *any* of the steps. This suggests that overall, Wikipedia is a good corpus for questions in StrategyQA that were written independently of the context.

Do matched paragraphs provide evidence? We assess the quality of matched paragraphs by analyzing both example-level and step-level annotations. First, we sample 217 decomposition steps with their corresponding paragraphs matched by one of the three workers. We let 3 different crowdworkers decide whether the paragraphs provide evidence for the answer to that step. We find that in 93% of the cases, the majority vote is that the evidence is valid.[4]

Next, we analyze annotations of the verification task (§3.4), where workers are asked to answer all decomposition steps based only on the matched paragraphs. We find that the workers could answer sub-questions and derive the correct answer in 82 out of 100 annotations. Moreover, in 6 questions indeed there was an error in evidence matching, but another worker who annotated the example was able to compensate for the error, leading to 88% of the questions where evidence matching succeeds. In the last 12 cases indeed evidence is missing, and is possibly absent from Wikipedia.

Lastly, we let experts review the paragraphs matched by one of the three workers to all the decomposition steps of a question, for 100 random questions. We find that for 79 of the questions the matched paragraphs provide sufficient evidence for answering the question. For 12 of the 21 questions without sufficient evidence, the experts indicated they would expect to find evidence in Wikipedia, and the worker probably could not find it. For the remaining 9 questions, they estimated that evidence is probably absent from Wikipedia.

In conclusion, 93% of the paragraphs matched at the step-level were found to be valid. Moreover, when considering single-worker annotations, ∼80% of the questions are matched with paragraphs that provide sufficient evidence for all retrieval steps. This number increases to 88% when aggregating the annotations of three workers.

Do different annotators match the same evidence paragraphs? To compare the evidence paragraphs matched by different workers, we check whether for a given decomposition step, the same paragraph IDs are retrieved by different annotators. Given two non-empty sets of paragraph IDs $\mathcal{P}_1, \mathcal{P}_2$, annotated by two workers, we compute the Jaccard coefficient $J(\mathcal{P}_1, \mathcal{P}_2) = \frac{|\mathcal{P}_1 \cap \mathcal{P}_2|}{|\mathcal{P}_1 \cup \mathcal{P}_2|}$. In addition, we take the sets of corresponding Wikipedia page IDs $\mathcal{T}_1, \mathcal{T}_2$ for the matched paragraphs, and compute $J(\mathcal{T}_1, \mathcal{T}_2)$. Note that a score of 1 is given to two identical sets, while a score of 0 corresponds to sets that are disjoint. The average similarity score is 0.43 for paragraphs and 0.69 for pages. This suggests that evidence for a decomposition step can be found in more than one paragraph in the same page, or in different pages.

4.3 Data Diversity

We aim to generate creative and diverse questions. We now analyze diversity in terms of the required reasoning skills and question topic.

Reasoning Skills To explore the required reasoning skills in StrategyQA, we sampled 100 examples and let two experts (authors) discuss and annotate each example with a) the type of strategy for decomposing the question, and b) the required reasoning and knowledge

[4]With moderate annotator agreement of $\kappa = 0.42$.

Strategy	Example	%
Physical	*Can human nails carve a statue out of quartz?*	13
Biological	*Is a platypus immune from cholera?*	11
Historical	*Were mollusks an ingredient in the color purple?*	10
Temporal	*Did the 40th president of the United States forward lolcats to his friends?*	10
Definition	*Are quadrupeds represented on Chinese calendar?*	8
Cultural	*Would a compass attuned to Earth's magnetic field be a bad gift for a Christmas elf?*	5
Religious	*Was Hillary Clinton's deputy chief of staff in 2009 baptised?*	5
Entertainment	*Would Garfield enjoy a trip to Italy?*	4
Sports	*Can Larry King's ex-wives form a water polo team?*	4

Table 6: Top strategies in StrategyQA and their frequency in a 100 example subset (accounting for 70% of the analyzed examples).

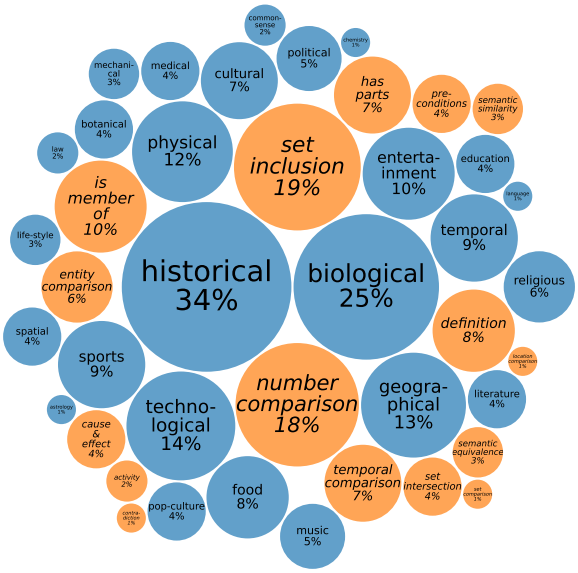

Figure 4: Reasoning skills in StrategyQA; each skill is associated with the proportion of examples it is required for. Domain-related and logical reasoning skills are marked in blue and orange (*italic*), respectively.

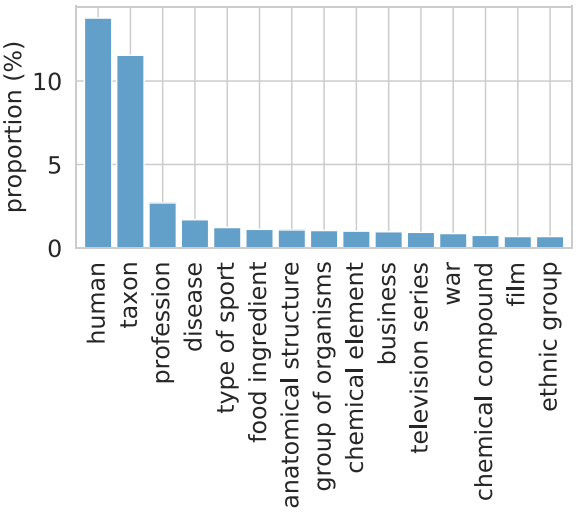

Figure 5: The top 15 categories of terms used to prime workers for question writing and their proportion.

skills per decomposition step. We then aggregate similar labels (e.g., botanical → biological) and compute the proportion of examples each strategy/reasoning skill is required for (an example can have multiple strategy labels).

Table 6 demonstrates the top strategies, showing that StrategyQA contains a broad set of strategies. Moreover, diversity is apparent (Figure 4) in terms of both domain-related reasoning (e.g., biological and technological) and logical functions (e.g., set inclusion and ''is member of''). While the reasoning skills sampled from questions in StrategyQA do not necessarily reflect their prevalence in a ''natural'' distribution, we argue that promoting research on methods for inferring strategies is an important research direction.

Question Topics As questions in StrategyQA were triggered by Wikipedia terms, we use the ''instance of'' Wikipedia property to characterize the topics of questions.[5] Figure 5 shows the distribution of topic categories in StrategyQA. The distribution shows StrategyQA is very diverse, with the top two categories (''human''

[5]It is usually a 1-to-1 mapping from a term to a Wikipedia category. In cases of 1-to-many, we take the first category.

and ‘‘taxon’’; i.e., a group of organisms) covering only a quarter of the data, and a total of 609 topic categories.

We further compare the diversity of StrategyQA to HotpotQA, a multi-hop QA dataset over Wikipedia paragraphs. To this end, we sample 739 pairs of evidence paragraphs associated with a single question in both datasets, and map the pair of paragraphs to a pair of Wikipedia categories using the ‘‘instance of’’ property. We find that there are 571 unique category pairs in StrategyQA, but only 356 unique category pairs in HotpotQA. Moreover, the top two category pairs in both of the datasets (‘‘human-human’’, ‘‘taxon-taxon’’) constitute 8% and 27% of the cases in StrategyQA and HotpotQA, respectively. This demonstrates the creativity and breadth of category combinations in StrategyQA.

4.4 Human Performance

To see how well humans answer strategy questions, we sample a subset of 100 questions from StrategyQA and have experts (authors) answer questions, given access to Wikipedia articles and an option to reveal the decomposition for every question. In addition, we ask them to provide a short explanation for the answer, the number of searches they conducted to derive the answer, and to indicate whether they have used the decomposition. We expect humans to excel at coming up with strategies for answering questions. Yet, humans are not necessarily an upper bound because finding the relevant paragraphs is difficult and could potentially be performed better by machines.

Table 7 summarizes the results. Overall, humans infer the required strategy and answer the questions with high accuracy. Moreover, the low number of searches shows that humans leverage background knowledge, as they can answer some of the intermediate steps without search. An error analysis shows that the main reason for failure (10%) is difficulty to find evidence, and the rest of the cases (3%) are due to ambiguity in the question that could lead to the opposite answer.

Answer accuracy	87%
Strategy match	86%
Decomposition usage	14%
Average # searches	1.25

Table 7: Human performance in answering questions. Strategy match is computed by comparing the explanation provided by the expert with the decomposition. Decomposition usage and the number of searches are computed based on information provided by the expert.

5 Experimental Evaluation

In this section, we conduct experiments to answer the following questions: a) How well do pre-trained language models (LMs) answer strategy questions? b) Is retrieval of relevant context helpful? and c) Are decompositions useful for answering questions that require implicit knowledge?

5.1 Baseline Models

Answering strategy questions requires external knowledge that cannot be obtained by training on StrategyQA alone. Therefore, our models and online solvers (§3.1) are based on pre-trained LMs, fine-tuned on auxiliary datasets that require reasoning. Specifically, in all models we fine-tune RoBERTa (Liu et al., 2019) on a subset of:

- BoolQ (Clark et al., 2019): A dataset for Boolean question answering.
- MNLI (Williams et al., 2018): A large natural language inference (NLI) dataset. The task is to predict if a textual premise entails, contradicts, or is neutral with respect to the hypothesis.
- Twenty Questions (20Q): A collection of 50K short commonsense Boolean questions.[6]
- DROP (Dua et al., 2019): A large dataset for numerical reasoning over paragraphs.

Models are trained in two configurations:

- **No context** : The model is fed with the question only, and outputs a binary prediction using the special `CLS` token.
- **With context** : We use BM25 (Robertson et al., 1995) to retrieve context from our corpus, while removing stop words from all queries. We examine two retrieval methods: a) question-based retrieval: by using the question as a query and taking the top

[6] https://github.com/allenai/twentyquestions.

$k = 10$ results, and b) decomposition-based retrieval: by initiating a separate query for each (gold or predicted) decomposition step and concatenating the top $k = 10$ results of all steps (sorted by retrieval score). In both cases, the model is fed with the question concatenated to the retrieved context, truncated to 512 tokens (the maximum input length of RoBERTa), and outputs a binary prediction.

Predicting Decompositions We train a seq-to-seq model, termed $\text{BART}_{\text{DECOMP}}$, that, given a question, generates its decomposition token-by-token. Specifically, we fine-tune BART (Lewis et al., 2020) on StrategyQA decompositions.

Baseline Models As our base model, we train a model as follows: We take a RoBERTa (Liu et al., 2019) model and fine-tune it on DROP, 20Q and BoolQ (in this order). The model is trained on DROP with multiple output heads, as in Segal et al. (2020), which are then replaced with a single Boolean output.[7] We call this model RoBERTa*.

We use RoBERTa* and RoBERTa to train the following models on StrategyQA: without context ($\text{RoBERTa*}_{\varnothing}$), with question-based retrieval ($\text{RoBERTa*}_{\text{IR-Q}}$, $\text{RoBERTa}_{\text{IR-Q}}$), and with predicted decomposition-based retrieval ($\text{RoBERTa*}_{\text{IR-D}}$).

We also present four oracle models:

- $\text{RoBERTa*}_{\text{ORA-P}}$: Uses the gold paragraphs (no retrieval).
- $\text{RoBERTa*}_{\text{IR-ORA-D}}$: Performs retrieval with the gold decomposition.
- $\text{RoBERTa*}^{\text{last-step}}_{\text{ORA-P-D}}$: Exploits both the gold decomposition and the gold paragraphs. We fine-tune RoBERTa on BoolQ and SQuAD (Rajpurkar et al., 2016) to obtain a model that can answer single-step questions. We then run this model on StrategyQA to obtain answers for all decomposition sub-questions, and replace all placeholder references with the predicted answers. Last, we fine-tune RoBERTa* to answer the last decomposition step of StrategyQA, for which we have supervision.
- $\text{RoBERTa*}^{\text{last-step-raw}}_{\text{ORA-P-D}}$: RoBERTa* that is fine-tuned to predict the answer from the gold paragraphs and the last step of the gold decomposition, *without* replacing placeholder references.

[7] For brevity, exact details on model training and hyper-parameters will be released as part of our codebase.

Model	Solver group(s)
$\text{RoBERTa}_{\varnothing}$(20Q)	Ptd, Fntd
$\text{RoBERTa}_{\varnothing}$(20Q+BoolQ)	Ptd, Fntd
$\text{RoBERTa}_{\varnothing}$(BoolQ)	Ptd, Fntd
$\text{RoBERTa}_{\text{IR-Q}}$(BoolQ)	Ptd
$\text{RoBERTa}_{\text{IR-Q}}$(MNLI+BoolQ)	Ptd

Table 8: QA models used as online solvers during data collection (§3.1). Each model was fine-tuned on the datasets mentioned in its name.

Model	Accuracy	Recall@10
Majority	53.9	-
$\text{RoBERTa*}_{\varnothing}$	63.6 ± 1.3	-
$\text{RoBERTa}_{\text{IR-Q}}$	53.6 ± 1.0	0.174
$\text{RoBERTa*}_{\text{IR-Q}}$	63.6 ± 1.0	0.174
$\text{RoBERTa*}_{\text{IR-D}}$	61.7 ± 2.2	0.195
$\text{RoBERTa*}_{\text{IR-ORA-D}}$	62.0 ± 1.3	0.282
$\text{RoBERTa*}_{\text{ORA-P}}$	70.7 ± 0.6	-
$\text{RoBERTa*}^{\text{last-step-raw}}_{\text{ORA-P-D}}$	65.2 ± 1.4	-
$\text{RoBERTa*}^{\text{last-step}}_{\text{ORA-P-D}}$	72.0 ± 1.0	-

Table 9: QA accuracy (with standard deviation across 7 experiments), and retrieval performance, measured by Recall@10, of baseline models on the test set.

Online Solvers For the solvers integrated in the data collection process (§3.1), we use three no-context models and two question-based retrieval models. The solvers are listed in Table 8.

5.2 Results

Strategy QA performance Table 9 summarizes the results of all models (§5.1). $\text{RoBERTa*}_{\text{IR-Q}}$ substantially outperforms $\text{RoBERTa}_{\text{IR-Q}}$, indicating that fine-tuning on related auxiliary datasets before StrategyQA is crucial. Hence, we focus on RoBERTa* for all other results and analysis.

Strategy questions pose a combined challenge of retrieving the relevant context, and deriving the answer based on that context. Training

without context shows a large accuracy gain of $53.9 \rightarrow 63.6$ over the majority baseline. This is far from human performance, but shows that some questions can be answered by a large LM fine-tuned on related datasets without retrieval. On the other end, training with *gold* paragraphs raises performance to 70.7. This shows that high-quality retrieval lets the model effectively reason over the given paragraphs. Last, using both gold decompositions and retrieval further increases performance to 72.0, showing the utility of decompositions.

Focusing on retrieval-based methods, we observe that question-based retrieval reaches an accuracy of 63.6 and retrieval with gold decompositions results in an accuracy of 62.0. This shows that the quality of retrieval even with gold decompositions is not high enough to improve the 63.6 accuracy obtained by RoBERTA*$_{\varnothing}$, a model that uses no context. Retrieval with predicted decompositions results in an even lower accuracy of 61.7. We also analyze predicted decompositions below.

Retrieval Evaluation A question decomposition describes the reasoning steps for answering the question. Therefore, using the decomposition for retrieval may help obtain the relevant context and improve performance. To test this, we directly compare performance of question- and decomposition-based retrieval with respect to the annotated gold paragraphs. We compute Recall@10, that is, the fraction of the gold paragraphs retrieved in the top-10 results of each method. Since there are 3 annotations per question, we compute Recall@10 for each annotation and take the maximum as the final score. For a fair comparison, in decomposition-based retrieval, we use the top-10 results across *all* steps.

Results (Table 9) show that retrieval performance is low, partially explaining why retrieval models do not improve performance compared to RoBERTA*$_{\varnothing}$, and demonstrating the retrieval challenge in our setup. Gold decomposition-based retrieval substantially outperforms question-based retrieval, showing that using the decomposition for retrieval is a promising direction for answering multi-step questions. Still, predicted decomposition-based retrieval does not improve retrieval compared to question-based retrieval, showing better decomposition models are needed.

To understand the low retrieval scores, we analyzed the query results of 50 random decomposition steps. Most failure cases are due to the shallow pattern matching done by BM25—for example, failure to match synonyms. This shows that indeed there is little word overlap between decomposition steps and the evidence, as intended by our pipeline design. In other examples, either a key question entity was missing because it was represented by a reference token, or the decomposition step had complex language, leading to failed retrieval. This analysis suggests that advances in neural retrieval might be beneficial for StrategyQA.

Human Retrieval Performance To quantify human performance in finding gold paragraphs, we ask experts to find evidence paragraphs for 100 random questions. For half of the questions we also provide decomposition. We observe average Recall@10 of 0.586 and 0.513 with and without the decomposition, respectively. This shows that humans significantly outperform our IR baselines. However, humans are still far from covering the gold paragraphs, since there are multiple valid evidence paragraphs (§4.2), and retrieval can be difficult even for humans. Lastly, using decompositions improves human retrieval, showing decompositions indeed are useful for finding evidence.

Predicted Decompositions Analysis shows that BART$_{\text{DECOMP}}$'s decompositions are grammatical and well-structured. Interestingly, the model generates strategies, but often applies them to questions incorrectly. For example, the question *Can a lifeboat rescue people in the Hooke Sea?* is decomposed to *1) What is the maximum depth of the Hooke Sea? 2) How deep can a lifeboat dive? 3) Is #2 greater than or equal to #1?*. While the decomposition is well-structured, it uses a wrong strategy (lifeboats do not dive).

6 Related Work

Prior work has typically let annotators write questions based on an entire context (Khot et al., 2020a; Yang et al., 2018; Dua et al., 2019; Mihaylov et al., 2018; Khashabi et al., 2018). In this work, we prime annotators with minimal information (few tokens) and let them use their

imagination and own wording to create questions. A related priming method was recently proposed by Clark et al. (2020), who used the first 100 characters of a Wikipedia page.

Among multi-hop reasoning datasets, our dataset stands out in that it requires *implicit* decompositions. Two recent datasets (Khot et al., 2020a; Mihaylov et al., 2018) have considered questions requiring implicit facts. However, they are limited to specific domain strategies, while in our work we seek diversity in this aspect.

Most multi-hop reasoning datasets do not fully annotate question decomposition (Yang et al., 2018; Khot et al., 2020a; Mihaylov et al., 2018). This issue has prompted recent work to create question decompositions for existing datasets (Wolfson et al., 2020), and to train models that generate question decompositions (Perez et al., 2020; Khot et al., 2020b; Min et al., 2019). In this work, we annotate question decompositions as part of the data collection.

7 Conclusion

We present StrategyQA, the first dataset of *implicit* multi-step questions requiring a wide-range of reasoning skills. To build StrategyQA, we introduced a novel annotation pipeline for eliciting creative questions that use simple language, but cover a challenging range of diverse strategies. Questions in StrategyQA are annotated with decomposition into reasoning steps and evidence paragraphs, to guide the ongoing research towards addressing implicit multi-hop reasoning.

Acknowledgments

We thank Tomer Wolfson for helpful feedback and the REVIZ team at Allen Institute for AI, particularly Michal Guerquin and Sam Skjonsberg. This research was supported in part by the Yandex Initiative for Machine Learning, and the European Research Council (ERC) under the European Union Horizons 2020 research and innovation programme (grant ERC DELPHI 802800). Dan Roth is partly supported by ONR contract N00014-19-1-2620 and DARPA contract FA8750-19-2-1004, under the Kairos program. This work was completed in partial fulfillment for the PhD degree of Mor Geva.

References

Max Bartolo, Alastair Roberts, Johannes Welbl, Sebastian Riedel, and Pontus Stenetorp. 2020. Beat the AI: Investigating adversarial human annotation for reading comprehension. *Transactions of the Association for Computational Linguistics*, 8:662–678. **DOI:** https://doi.org/10.1162/tacl_a_00338

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. 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 2924–2936, Minneapolis, Minnesota. 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. *Transactions of the Association for Computational Linguistics (TACL)*, 8:454–470. **DOI:** https://doi.org/10.1162/tacl_a_00317

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. 2020. ERASER: A benchmark to evaluate rationalized NLP models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4443–4458. Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1/2020.acl-main.408

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Mor Geva, Yoav Goldberg, and Jonathan Berant. 2019. Are we modeling the task or the

annotator? An investigation of annotator bias in natural language understanding datasets. 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 1161–1166, Hong Kong, China. Association for Computational Linguistics.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. 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 107–112, New Orleans, Louisiana. Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1/N18-2017

Yichen Jiang and Mohit Bansal. 2019. Avoiding reasoning shortcuts: Adversarial evaluation, training, and model development for multi-hop QA. In *Association for Computational Linguistics (ACL)*. **DOI:** https://doi.org/10.18653/v1/P19-1262, **PMID:** 31353678

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262, New Orleans, Louisiana. Association for Computational Linguistics, **DOI:** https://doi.org/10.18653/v1/N18-1023

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. 2020a. QASC: A dataset for question answering via sentence composition. In *AAAI*. **DOI:** https://doi.org/10.1609/aaai.v34i05.6319

Tushar Khot, Daniel Khashabi, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2020b. Text modular networks: Learning to decompose tasks in the language of existing models. *arXiv preprint arXiv:2009.00751*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics (TACL)*, 7:453–466. **DOI:** https://doi.org/10.1162/tacl_a_00276

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. 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. Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1/2020.acl-main.703

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 preprint arXiv:1907.11692*.

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 *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1/D18-1260

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

Florence, Italy. Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1/P19-1613

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Workshop on Cognitive Computing at NIPS*.

Ethan Perez, Patrick Lewis, Wen-tau Yih, Kyunghyun Cho, and Douwe Kiela. 2020. Unsupervised question decomposition for question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8864–8880.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*. **DOI:** https://doi.org/10.18653/v1/D16-1264

Stephen Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. 1995. Okapi at TREC-3. In *Overview of the Third Text REtrieval Conference (TREC-3)*, pages 109–126. Gaithersburg, MD: NIST.

Elad Segal, Avia Efrat, Mor Shoham, Amir Globerson, and Jonathan Berant. 2020. A simple and effective model for answering multi-span questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3074–3080. **DOI:** https://doi.org/10.18653/v1/2020.emnlp-main.248

Alane Suhr, Stephanie Zhou, Ally Zhang, Iris Zhang, Huajun Bai, and Yoav Artzi. 2019. A corpus for reasoning about natural language grounded in photographs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6418–6428, Florence, Italy. Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1/P19-1644

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *North American Chapter of the Association for Computational Linguistics (NAACL)*. **DOI:** https://doi.org/10.18653/v1/N18-1059

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics (TACL)*, 6:287–302. **DOI:** https://doi.org/10.1162/tacl_a_00021

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana, Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1/N18-1101

Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. 2020. Break it down: A question understanding benchmark. *Transactions of the Association for Computational Linguistics (TACL)*, **DOI:** https://doi.org/10.1162/tacl_a_00309

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 *Empirical Methods in Natural Language Processing (EMNLP)*, **DOI:** https://doi.org/10.18653/v1/D18-1259, **PMCID:** PMC6156886