ConditionalQA: A Complex Reading Comprehension Dataset with Conditional Answers

Haitian Sun
School of Computer Science
Carnegie Mellon University
haitians@cs.cmu.edu

William W. Cohen
Google Research
wcohen@google.com

Ruslan Salakhutdinov
School of Computer Science
Carnegie Mellon University
rsalakhu@cs.cmu.edu

Abstract

We describe a Question Answering (QA) dataset that contains complex questions with *conditional answers*, i.e. the answers are only applicable when certain conditions apply. Answering the questions requires compositional logical reasoning across complex context. We call this dataset ConditionalQA. In addition to conditional answers, the dataset also features: (1) long context documents with information that is related in logically complex ways; (2) multi-hop questions that require compositional logical reasoning; (3) a combination of extractive questions, yes/no questions, questions with multiple answers, and not-answerable questions; (4) questions asked without knowing the answers. We show that ConditionalQA is challenging for many of the existing QA models, especially in selecting answer conditions. We believe that this dataset will motivate further research in understanding complex documents to answer hard questions.[1]

1 Introduction

Many reading comprehension (RC) datasets have been recently proposed (Rajpurkar et al., 2016, 2018; Kwiatkowski et al., 2019; Yang et al., 2018; Dasigi et al., 2021; Ferguson et al., 2020). In a reading comprehension task, models are provided with a document and a question and asked to find the answers. Questions in existing reading comprehension datasets generally have a unique answer or a list of answers that are equally correct, e.g. "Who was the president of the US?" with the answers "George Washington", "Thomas Jefferson", etc. We say that these questions have *deterministic* answers. However, questions in the real world do not always have deterministic answers, i.e. answers to the questions are different under different conditions. For example, in Figure 1, the document discusses "Funeral Expense Payment" and a question asks an applicant's eligibility. This question cannot be deterministically answered: the answer is "yes" only if "you're arranging a funeral in the UK", while the answer is "no", if "... another close relative of the deceased is in work" is true. We call answers that are different under different conditions *conditional answers*.

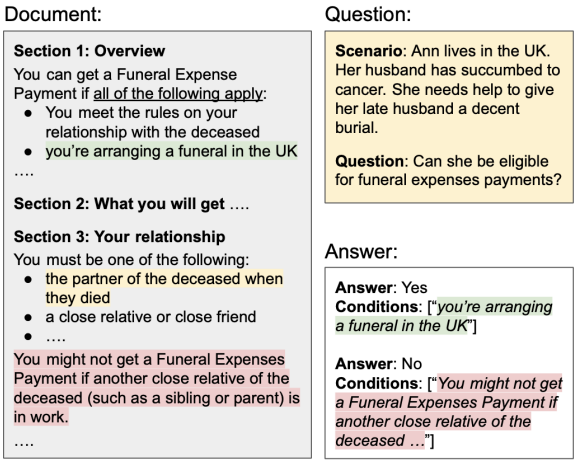

Figure 1: An example of question and document in ConditionalQA dataset. The left side is a snapshot of the document discussing the eligibility of the benefit "Funeral Expense Payment". The text span "Her husband" satisfies the requirement on the "relationship with the decease" (in yellow). Text pieces in green and red are requirements that must be satisfied and thus selected as conditions for the "Yes" and "No" answers.

A conditional answer consists of an *answer* and a list of *conditions*. An answer is only true if its conditions apply. In the example above, "you are arranging a funeral in the UK" is the condition for the answer "yes". An answer can have multiple conditions. Conditional answers are commonly seen when the context so complex so asking a *complete* question with a deterministic answer is impractical; for example, when a person asks a question with some prior knowledge in mind but cannot enumerate all necessary details. A practical way to answer incomplete questions is to find all possible answers to the question – and if some answers are only true

[1] https://haitian-sun.github.io/conditionalqa/

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 3627 - 3637

under certain conditions, the conditions should be output as well. Answering such questions generally requires the models to understand the complex logic in the context and perform extensive reasoning to identify the answers and conditions.

We present the ConditionalQA dataset, which contains questions with conditional answers. We take documents from the UK government website[2] as our corpus. Documents in this corpus discuss public policies in the UK and were first used in the ShARC dataset (Saeidi et al., 2018). It is partically interesting for constructing the ConditionalQA dataset because it contains complex contents with complex internal logic such as conjunction, disjunction, and exception (see the example in Figure 1). Questions in ConditionalQA are asked by human annotators. Each example contains a question, a scenario when the question is asked, and a document that discusses the policy that the question asks about. The task is to find all possible answers to the questions that apply to the user's scenario. If an answer is only true under certain conditions, the model should return the list of conditions along with the answer. Answers and conditions are annotated by human annotators with the exact input, i.e. the question, the scenario, and the associated document. We provide supporting evidences labeled by human annotators as additional supervision.

In addition to having conditional answers, ConditionalQA also features the following properties. First, the documents in ConditionalQA have complex structure. As opposed to Wikipedia pages, where most sentences or paragraphs contain standalone information, documents in ConditionalQA usually have complex internal logic that is crucial for answering the questions. Second, many questions in the dataset are naturally multi-hop, as illustrated in the example on Figure 1, e.g. being "the partner of the deceased" satisfied the requirement on "your relationship with the deceased" which is one of high-level requirements to obtain the benefit. Answering those question requires models that understand the internal logic within the document and reason over the it to find correct answers. Third, we decouple the asking and answering process when annotating questions, as suggested by Ferguson et al. (2020); Dasigi et al. (2021); Clark et al. (2020), so questions are asked without knowing the answers. Forth, ConditionalQA contains various types of questions including yes/no questions and extractive questions. Questions can have one or multiple answers, or can be not answerable, as a result of the decoupled annotation process.

We experimented with several strong baseline models on ConditionalQA (Ainslie et al., 2020; Sun et al., 2021; Izacard and Grave, 2021). The best performing model achieves only 64.9% accuracy on yes/no questions, marginally better than the majority baseline (62.2% if always predicting "yes"), and 25.2% exact match (EM) on extractive answers. We further measure the accuracy of jointly predicting answers and conditions, in which case the accuracy drops to 49.1% and 22.5%. The best metrics with conditions are obtained if *no condition* is predicted, showing how challenging it is for existing models to predict conditional answers.

2 Related Works

Many question answering datasets have been proposed in the past few years (Rajpurkar et al., 2016, 2018; Yang et al., 2018; Dasigi et al., 2021; Ferguson et al., 2020; Kwiatkowski et al., 2019) and research on these has significantly boosted the performance of QA models. As large pretrained language models (Devlin et al., 2019; Liu et al., 2019; Ainslie et al., 2020; Beltagy et al., 2020; Guu et al., 2020; Verga et al., 2020) achieved better performance on traditional reading comprehension and question answering tasks, efforts have been made to make the questions more complex. Several multi-hop QA datasets were released (Yang et al., 2018; Ferguson et al., 2020; Talmor and Berant, 2018; Welbl et al., 2018) to test models' ability to solve complex questions. However, most questions in these datasets are answerable by focusing on a small piece of evidence at a time, e.g. a sentence or a short passage, leaving reasoning through long and complex contents a challenging but unsolved problem.

Some datasets have been recently proposed for question answering over long documents. QASPER (Dasigi et al., 2021) contains questions asked from academic papers, e.g. "What are the datasets experimented in this paper?". To answer those questions, the model should read several sections and collect relevant information. NarrativeQA (Mou et al., 2021) requires reading entire books or movie scripts to answer questions about their characters or plots. Other datasets, e.g. HybridQA (Chen et al., 2021b), can also be viewed as question answering over long documents if tables with hyper-linked text from the cells are flattened into

[2] https://www.gov.uk/parental-leave

a hierarchical document. ShARC (Saeidi et al., 2018) is a conversational QA dataset that also use UK government websites as its corpus. However, the ShARC dataset only contains yes/no questions and the conversation history is generated by annotators with the original rule text in hand, making the conversation artificial. The length of context in ShARC is usually short, such as a few sentences or a short paragraph. While using the same corpus, ConditionalQA contains completely different questions and new types of answers. It focuses on a new problem that has not been previously studied.

Most of the existing datasets, including the ones discussed above, contain questions with unique answers. Answers are unique because questions are well specified, e.g. "Who is the president of the US in 2010?". However, questions can be ambiguous if not all information is provided in the question, e.g. "When was the Harry Potter movie released?" does not specify which Harry Potter movie. AmbigQA (Min et al., 2020) contains questions that are ambiguous, and requires the model to find all possible answers of an ambiguous question and rewrite the question to make it well specified. Similar datasets Temp-LAMA (Dhingra et al., 2021), TimeQA (Chen et al., 2021a) and SituatedQA (Zhang and Choi, 2021) have been proposed that include questions that require resolving temporal or geographic ambiguity in the context to find the answers. They are similar to ConditionalQA in that questions are incomplete, but ConditionalQA focuses on understanding documents with complex logic and answering questions with conditions. It's usually not possible to disambiguate questions in ConditionalQA as rewriting the questions (or scenarios) to reflect all conditions of answers to make the questions deterministic is impractical.

We create ConditionalQA in the public policy domain. There are some existing domain specific datasets, including PubMedQA and BioAsq (Nentidis et al., 2018; Jin et al., 2019) in medical domain, UDC (Lowe et al., 2016) in computer software domain, QASPER (Dasigi et al., 2021) in academic paper domain, PrivacyQA and PolicyQA (Ahmad et al., 2020; Ravichander et al., 2019) in legal domain, etc. PrivacyQA and PolicyQA have similar context as ConditionalQA, but the questions do not require compositional reasoning and the answers are short text spans. We use a corpus in the public policy domain because it is easy to understand by non-experts while being complex enough to support challenging questions. ConditionalQA is not designed to be a domain specific dataset.

3 The Task

In our task, the model is provided with a long document that describes a public policy, a question about this document, and a user scenario. The model is asked to read the document and find all answers and their conditions if any.

3.1 Corpus

Documents in ConditionalQA describe public policies in the UK, e.g. "Apply for Visitor Visa" or "Punishment of Driving Violations". Each document covers a unique topic and the contents are grouped into sections and subsections. Contents in the same section are closely related but may also be referred in other sections. We create ConditionalQA in this domain because these documents are rather complex with internal logic, yet annotators are familiar with the content so they can ask natural yet challenging questions, compared to formal legal or financial documents with more sophisticated terms and language.

3.2 Input

The input to a reading comprehension model consists of a document, a question, and a user scenario:

- A *document* describes a public policy in the UK. Content of a document is coherent and hierarchical, structured into sections and subsections. Documents are crawled from the website and processed by serializing the DOM trees of the web pages into lists of HTML elements with tags, such as <h1>, <p>, <li>, and <tr>. Please see more information in §4.1.
- A *question* asks about a specific aspect of the document, such as eligibility or other aspects with "how", "when", "what", "who", "where", etc. Questions are relevant to the content in the document, even though they may be "not answerable".
- A *user scenario* provides background information for the question. Some information will be used to restrict the answers that can be possibly correct. Not all information in the user scenario is relevant because they are written by crowd source workers without seeing the full document or knowing the answers. Information in the scenario is also likely to be incomplete. This setup simulates the real

information seeking process of having both irrelevant and incomplete information.

3.3 Output

A reading comprehension model is asked to predict the answers and the list of conditions if there is any.

- An *answer* to the question has three different types: (1) "yes" or "no" for questions such as "Can I get this benefit?"; (2) an extracted text span for questions asking "how", "when", "what", etc.; (3) "not answerable" if an answer does not exist in the document. Since the information to get a definite answer is sometimes incomplete, besides predicting the answers, the model is asked to identify their conditions.

- A *condition* contains information that must be satisfied in order to make the answer correct but is not mentioned in the user scenario. In ConditionalQA, we restrict a condition to be one of the HTML elements in the document instead of the exact extracted text span.[3] Selected conditions are then evaluated as a retrieval task with F1 at the element level, i.e. the model should retrieve all HTML elements with unsatisfied information to get a perfect F1 score. If no condition is required, the model must return an empty list. Please see §3.4 for more details on evaluation.

3.4 Evaluation

We evaluate performance of models on the ConditionalQA dataset as a reading comprehension (RC) task. Answers are measured with exact match (EM) and F1. Some questions have multiple answers. The model should correctly predict all possible answers to get the full score. Since the order of answers does not matter, to compute the metrics, we compare all possible permutations of the predicted answers to the list of correct answers. We take the best result among all permutations as the result for this example. Let $\{\hat{a}_1, \ldots, \hat{a}_m\}$ be the list of predicted answer and $\{a_1, \ldots, a_n\}$ the reference answers. The EM of the predicted answers is

$$\text{EM} = \max_{\{\tilde{a}_1,\ldots,\tilde{a}_m\}} \frac{1}{n} \sum_{i=1}^{\min(m,n)} s_{em}(\tilde{a}_i, a_i) \cdot \gamma_{m,n} \quad (1)$$

$$\gamma_{m,n} = \begin{cases} e^{1-m/n} & \text{if } m > n \\ 1 & \text{if } m \leq n \end{cases}$$

where $\{\tilde{a}_1, \ldots, \tilde{a}_m\}$ is a permutation of the predicted answers $\{\hat{a}_1, \ldots, \hat{a}_m\}$ and $s_{em}(\cdot, \cdot)$ is the scoring function that measures EM between two text spans. $\gamma_{m,n}$ is a penalty term that is smaller than 1 if more answers than the reference answers are predicted, i.e. $m > n$. We compute token-level F1 in the similar way using the scoring function $s_{f1}(\cdot, \cdot)$ on the extracted answer spans. For not answerable questions, EM and F1 are 1.0 if and only if no answer is predicted.

We additionally measure the performance of answers with conditions. We adopt the same permutation strategy as above, except that the scoring function will also take into account the accuracy of predicted conditions. Let $\hat{C}_i$ be the set of predicted conditions for the predicted answer $\hat{a}_i$ and C_i be the oracle conditions for the answer a_i. The new scoring function for the predicted answer with conditions is

$$s_{em+c}(\tilde{a}_i, \tilde{C}_i, a_i, C_i) = s_{em}(\tilde{a}_i, a_i) \cdot \text{F1}(\hat{C}_i, C_i)$$

where $\text{F1}(\cdot, \cdot)$ measures the accuracy of the set of predicted conditions at HTML element level. Recall that conditions are restricted to select from HTML elements in the document. $\text{F1}(\hat{C}_i, C_i)$ equals to 1 if and only if all required conditions are selected. This is different from $s_{f1}(\cdot, \cdot)$ that measures token level F1 of the extracted answers. If the answer does not require any conditions, the model should predict an empty set. We simply replace the scoring function $s_{em}(\cdot, \cdot)$ in Eq. 1 with $s_{em+c}(\cdot, \cdot)$ to compute EM with conditions.

4 Data Collection

4.1 Documents

Documents are originally presented on the UK government website in the HTML format. We crawled the pages from the website and processed it to only keep the crucial tags, that include:

- Headings <h1, h2, h3, h4>: We keep headings at different levels. This can be used to identify the hierarchical structure in the documents.
- Text <p>: This tag is used for general contents. We replace descriptive tags, e.g. , with the plain tag <p> for simplicity.
- List <li>: We keep the tags for list items, but drop their parent tags <ul> or <ol>. We observe that very few ordered lists (<ol>) have

[3] We argue that selecting HTML elements as conditions is already very challenging (see experimental results in §5.2) and leave extracting the exact text spans as future work.

been used in the dataset, so we will not distinguish them.

- Table <tr>: Again, we drop their parent tags <table> to simplify the document format. We further remove the <td> and <th> tags from cells and concatenate cells in the same row with the separation of " | ".

A processed document contains a list of strings that starts with a tag, follows with its content, and ends with the tag, e.g. ["<h1> Overview </h1>", " You can apply for ... ", . . .].

We drop some common sections that do not contain any crucial information, e.g. "How to Apply", to make sure that questions are specific to the topic of the documents. We further require that the document should contain at least 3 sections. We end up with 652 documents as our corpus. The max length of the documents is 9230 words (16154 sub-words in T5 (Raffel et al., 2020)).

4.2 Questions

We collect questions from crowd source workers on Amazon Mechanical Turk. To encourage workers asking questions not be restricted to a specific piece of text, we hide the full document but instead provide a snippet of the document to the workers. A snippet includes a table of content that contains section and subsection titles (from <h1> and <h2> tags), and the very first subsection in the document that usually provides a high level overview of the topic. The snippet lets workers get familiar with the topic of this document so they can ask closely relevant questions. We observe that restricting the geographic location of workers to the UK can significantly improve the quality of questions because local residents are more familiar with their policies.

We ask the workers to perform three sub-tasks when coming up with the questions. First, we ask the workers to provide three attributes that can identify the group of people who may benefit from or be regulated by the policy discussed in the document. Second, they are asked to come up with a scenario when they will want to read this document and a question about what they would like to know. Third, workers are asked to mark which attributes have been mentioned in their question and scenario. When assessing the annotation quality, we find that asking workers to provide attributes makes the questions and scenarios much more specific, significantly improving the quality of the dataset.

We assign 3 workers to documents with four or more sections and 2 workers to documents with three sections. Each worker is asked to give two questions and the two questions have to be diverse. We collect 3617 questions in this stage.

4.3 Find Answers

We hire another group of workers to work on the answer portion of this task. Finding answers is very challenging to crowd source workers because it requires the workers to read the full document carefully to understand every piece of information in the document. We provide one-on-one training for the workers to teach them how to select supporting evidences, answers, and conditions.

Workers are asked to perform three sub-tasks. The first step is to select supporting evidences from the document. Supporting evidences are HTML elements that are closely related to the questions, including elements that have content that directly justify the answers and the ones that will be selected as conditions in the next step. In the second step, workers are asked to type answers and select associated conditions. Workers can input as many answers as possible or mark the question as "not answerable". For each answer, they can select one or more supporting evidences as the answer's conditions if needed. Workers are asked not to select conditions if there is sufficient information in the scenario to answer the question. We give workers permission to slightly modify the questions or scenarios if the questions are not clearly stated, or they can mark it as a bad question (different from not answerable) so we will drop it from the dataset.

We additionally perform a revise step to improve the annotation quality. We provide the union of selected evidences and answers from multiple annotations of a question to an additional group of annotators and let them deselect unrelated evidences and merge answers. As the amount of information provided to workers at this step is significantly less than in the previous answer selection stage, the annotation quality improves significantly. We end up with 3102 questions with annotated answers.

4.4 Move Conditions to Scenario

To encourage the model of learning subtle difference in user scenarios that affects the answers and conditions, we create new questions by modifying existing questions with conditional answers by moving one of the conditions to their scenarios.

Type	Scenario & Question	Answer w/ [*Conditions*]
Single answer	**Scenario**: "My father has recently appealed for a traffic ticket." **Question**: "How long will it take to get a decision?"	• "4 weeks"
Single answer w/ conditions	**Scenario**: "I applied to cut down a tree on my land but it was rejected 20 days ago" **Question**: "Am I still able to appeal against it?"	• "yes" ["* You can appeal before the date the tree replacement notice comes into effect. *"]
Multiple answers	**Scenario**: "I will get my first paycheck tomorrow." **Question**: "What information should be on my pay split?"	• "earnings before and after any deductions" • "the amount of any deductions" • "the number of hours you worked"
Multiple yes/no w/ conditions	**Scenario**: "I am looking at buying a new build home. I am 26 and a first-time buyer." **Question**: "Am I eligible to get an Equity Loan?"	• "yes" ["* able to afford fees and interest *", "* sold by an eligible homebuilder *"] • "no" ["* You can not apply if you had any form of sharia mortgage finance *"]
Multiple extractive w/ conditions	**Scenario**: "I always walk my labrador in open spaces. I forgot to clean up his mess yesterday." **Question**: "How much can I be fined for this?"	• "$100" ["* $100 on the spot *"] • "up to $1000" ["* - up to $1,000 if it goes to court *"]
Multiple extractive one w/ condition	**Scenario**: "I am about to apply for a Parent of a Child Student Visa to stay with my child for a year in the UK" **Question**: "What documents are needed to apply for this visa?"	• "a current passport or other travel document" • "proof that you have enough fund" • "your tuberculosis (tb) test results" ["* your tuberculosis (TB) test results if you are from a country where you have to take the TB test *"]

Table 1: Example of questions in ConditionalQA. Text pieces that follows the answers in the brackets are [*conditions*]. Some answers are deterministically correct so they are not followed by conditions.

Specifically, we show the workers the original questions, scenarios, and the annotated answers and conditions. Evidences are also provided for workers to get them familiar with the background of the questions and reasoning performed to get the original answers. Workers are asked to pick one of the conditions and modify the original scenario to reflect this condition. The modified questions and scenarios are sent back to the answering stage to get their annotations. We randomly select a small portion of the questions that have conditional answers as inputs to this stage so as to not affect the original distribution of the dataset. We collected 325 additional examples from this stage.

4.5 Train / Dev / Test Splits

We partition the dataset by documents to prevent leaking information between questions from the same document. The dataset contains 436 documents and 2338 questions in the training set, 59 documents and 285 questions in the development set, and 136 documents and 804 questions in the test set. Please see Appendix A for more statistics on ConditionalQA.

5 Evaluation

5.1 Baselines

Evaluating existing models on ConditionalQA is challenging. In addition to predicting answers to questions, the ConditionalQA task also asks the model to find the answers' conditions if any of them applies. To the best of our knowledge, no existing model fits the purpose of this task. We modified three competitive QA models as baselines to the ConditionalQA dataset. In addition to the new form of answers, traditional reading comprehension models also face the challenge that the context of questions in ConditionalQA is too long to fit into the memory of many Transformer-based models like BERT (Devlin et al., 2019) and even ETC (Ainslie et al., 2020). The baseline models we implemented are described below.

ETC: ETC (Ainslie et al., 2020) is a pretrained Transformer-based language model that is designed for longer inputs (up to 4096 tokens). ETC achieved the state-of-the-art on several challenging tasks, e.g. HotpotQA and WikiHop (Yang et al., 2018; Welbl et al., 2018). Since ETC cannot fit the entire document (with up to 16154 tokens) into its memory, we cannot let ETC to jointly predict answers and conditions, we designed a two stage

	Yes / No		Extractive		Conditional		Overall	
	answer	w/ conds	answer	w/ conds	answer	w/ conds*	answer	w/ conds
majority	62.2 / 62.2	42.8 / 42.8	– / –	– / –	– / –	– / –	– / –	– / –
ETC	63.1 / 63.1	47.5 / 47.5	8.9 / 17.3	6.9 / 14.6	39.4 / 41.8	2.5 / 3.4	35.6 / 39.8	26.9 / 30.8
DocHopper	**64.9 / 64.9**	**49.1 / 49.1**	17.8 / 26.7	15.5 / 23.6	42.0 / 46.4	3.1 / 3.8	40.6 / 45.2	31.9 / 36.0
FiD	64.2 / 64.2	48.0 / 48.0	**25.2 / 37.8**	**22.5 / 33.4**	**45.2 / 49.7**	**4.7 / 5.8**	**44.4 / 50.8**	**35.0 / 40.6**
human	91.4 / 91.4	82.3 / 82.3	72.6 / 84.9	62.8 / 69.1	74.7 / 86.9	48.3 / 56.6	82.6 / 88.4	73.3 / 76.2

Table 2: Experiment results on ConditionalQA (EM / F1). Numbers are obtained by re-running the open-sourced codes of the baselines. "majority" reflects the accuracy of always predicting "yes" without conditions. *See discussion in text.

pipeline to run ETC on ConditionalQA.

In the first stage, ETC is trained as a normal reading comprehension model to predict answers from the document by jointly encoding the questions and documents. We adopt a sequential reading approach that reads one section at a time. The answer with the highest probability among all sections will be considered as the final answer. We append three special tokens "`yes`", "`no`", and "`not answerable`" for the yes/no and not answerable questions. Since it is not clear how to extract multiple answers with the Transformer-based extractive QA model, we restrict to the number of predicted answers to one. The second stage in the pipeline is to select conditions. Questions, answers, and documents are concatenated together into a single input for ETC. We then use the embeddings of global tokens for sentences in ETC to predict conditions. Since the number of conditions for the answer is unknown, we train the condition selection process with a binary classification target, by labeling each global token as positive or negative. The threshold of selecting conditions is a hyper-parameter.

DocHopper: DocHopper (Sun et al., 2021) is an iterative attention method that extends ETC for reading long documents to answer multi-hop questions. It reads the full documents at once and jointly predicts answers and conditions. The model iteratively attends to information at different levels in the document to gather evidences to predict the final answers. We modify the iterative process in DocHopper for the purpose of this task: specifically, DocHopper is trained to run three iterative attention steps: (1) attend to the supporting evidences; (2) attend to the sentence that contains the answer; and (3) attend to the conditions. Since the query vector in each attention step is updated with information from the previous steps, conditions attended at the third step are aware of the previously predicted answers. Unfortunately, DocHopper is still restricted to predicting one answer for each question. The condition selection step in DocHopper is also trained with binary classification loss. Different from the ETC pipeline, the three attention steps are jointly optimized.

FiD: FiD (Izacard and Grave, 2021) is a generative model with an encoder-decoder architecture. The encoder reads multiple contexts independently and generates their embeddings. The decoder attends to all embeddings of the context to generate the final answers. In this task, we train FiD to sequentially generate the answers with conditions, i.e. $[a_1, c_{11}, c_{12}, \ldots, a_2, c_{21}, c_{22}, \ldots]$ where $\{a_1, \ldots, a_n\}$ are the correct answers and $\{C_1, \ldots, C_n\}$ are their conditions, i.e., $c_{ij} \in C_i$ is the j'th condition for the answer a_i. If C_i is empty, the model is trained to predict "NA" as the only condition for the i'th answer. FiD can predict multiple answers as opposed to ETC and DocHopper.

Human We randomly sample 80 questions and ask human annotators to answer them. Annotators are provided with the full instructions and 10 additional annotated examples to clarify the task. We do not provide additional training to the annotators.

5.2 Results

Experiment results are shown in Table 2. We report the numbers on yes/no questions and extractive questions separately. The numbers in Table 2 show that the ConditionalQA task is very challenging—the performance of the best model on yes/no questions is 64.9% (marginally higher than always predicting the majority answer "yes"), and the performance on extractive questions is 25.2% EM. FiD has the best performance on extractive questions because FiD can predict multiple answers while ETC-pipeline and DocHopper only predict one.

The performance drops significantly if answers and conditions are jointly evaluated. The best performance on jointly evaluating answers and conditions ("w/ conditions") in Table 2 is only 49.1% for yes/no questions and 22.5% EM for extractive questions. Even worse, this best result is obtained when *no* condition is selected, i.e. the threshold

Error types	%	Examples	Correct answers	Predictions
Not answerable	7.6	"Am I eligible for a tax reduction?"	not_answerable	"yes"
Wrong answer type (yes/no vs. extractive)	4.2	"How can I check if this design has been registered?"	"ask the intellectual property office to search for you"	"no"
Wrong answer (yes/no)	19.5	"Will it be classed as a small vessel?"	"yes"	"no"
Wrong answer (extractive, right type)	20.3	"How many points will I receive on my license?"	"6"	"3"
Wrong answer (extractive, wrong type)	9.3	"What is the account number should I send the money to?"	"12001020"	"hmrc"
Correct answer w/ wrong conditions	14.4	"Can I still send simpler annual accounts as a micro-entity?"	"yes", ["*$316,000 or less on its balance sheet"*]	"yes", []
Partial answer	24.5	"What will not need to be repeated for each trip?"	"a microchip", "rabies vaccination"	"a microchip"

Table 3: Error analysis on the predictions of the best performed model (FiD). The percentage is the fraction of errors made in that category over all errors.

of selecting conditions is 1.0. The difficulty of selecting conditions is more obvious if we focus on the subset of questions that have at least one conditional answer. The accuracy drops by 90% if answers and conditions are jointly evaluated.[4]

We also study how the threshold on the confidence scores of selecting conditions affects the evaluation results. Results are shown in Figure 2. As we decrease the threshold for selecting conditions, the EM with conditions on the subset of questions that have conditional answers slightly improves, but the overall EM with conditions drops dramatically due to the false positive conditions. FiD is a generative model so we can not evaluate it in the same way. In our evaluation, predictions from the best performing FiD checkpoint also do not select any conditions.

Table 4 shows the best results on the subset of questions that have conditional answers. Hyperparameters are tuned on the subset of questions. We could possibly get better results on questions with conditional answers with threshold $\epsilon < 1.0$, but the improvement is still marginal.

5.3 Error Analysis

We manually check 200 examples in the prediction of the best performed model FiD and label the type of errors made. The numbers are shown in Table 3. The most errors are made when only a subset of correct answers is predicted. This is due to the fact that the model (FiD) has a tendency to predict one answer for each question. The second most common errors are made by predicting answers with the correct type but wrong value. Such errors are commonly made by reading comprehension models in many tasks. The model made a lot of errors in yes/no questions because they consist of around 50% of the questions. The model is good at distinguishing yes/no questions and extractive question as producing the wrong kind of answer only makes up of 4.2% of the errors.

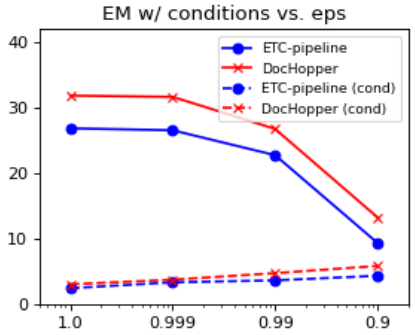

Figure 2: EM of answers with conditions with different thresholds of confidence (eps) on conditions. Dotted lines represent experiment results on the subset of questions that have conditional answers.

6 Conclusion

We propose a challenging dataset ConditionalQA that contains questions with conditional answers.

[4]The EM/F1 w/ conditions* is non-zero on this subset of questions even if no condition is ever selected, because some questions have both conditional and deterministic answers. Models get partial credits if they predicts the deterministic answers correctly.

	Best Overall	Best Conditional
ETC	2.5 / 3.4	4.4 / 4.6
DocHopper	3.1 / 3.8	**5.9 / 7.1**
FiD	**4.7 / 5.8**	4.7 / 5.8

Table 4: EM/F1 w/ conditions on the subset of questions with *conditional answers*. "Best Overall" uses the best checkpoints/hyper-parameters on the full dataset, while "Best Conditional" uses the best ones on the subset of questions.

The dataset requires models to understand complex logic in a document in order to find correct answers and conditions to the questions. Experiments on state-of-the-art QA models show that their overall performance on ConditionalQA is relatively poor. This also suggests that current QA models lack the reasoning ability to understand complex documents and answer hard questions with answers beyond single span extraction. We hope that this dataset will stimulate further research in building NLP models with better reasoning abilities.

7 Ethics Statements

This dataset should be ONLY used for NLP research purpose. Questions are artificial and do not contain any personal information. Answers are NOT provided by legal professionals and should NOT be used for any legal purposes.

8 Acknowledgement

This work was supported in part by the NSF IIS1763562, ONR Grant N000141812861, Google Research. We would also like to thank Vijay A. Saraswat <Vijay.Saraswat@gs.com> for valuable feedback.

References

Wasi Ahmad, Jianfeng Chi, Yuan Tian, and Kai-Wei Chang. 2020. PolicyQA: A reading comprehension dataset for privacy policies. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 743–749, Online. Association for Computational Linguistics.

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. 2020. Etc: Encoding long and structured inputs in transformers.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer.

Wenhu Chen, Xinyi Wang, and William Yang Wang. 2021a. A dataset for answering time-sensitive questions.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Wang. 2021b. Hybridqa: A dataset of multi-hop question answering over tabular and textual data.

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.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers.

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

Bhuwan Dhingra, Jeremy R. Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W. Cohen. 2021. Time-aware language models as temporal knowledge bases.

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

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

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William W. Cohen, and Xinghua Lu. 2019. Pubmedqa: A dataset for biomedical research question answering.

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

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.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2016. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. Ambigqa: Answering ambiguous open-domain questions.

Xiangyang Mou, Chenghao Yang, Mo Yu, Bingsheng Yao, Xiaoxiao Guo, Saloni Potdar, and Hui Su. 2021. Narrative question answering with cutting-edge open-domain qa techniques: A comprehensive study.

Anastasios Nentidis, Anastasia Krithara, Konstantinos Bougiatiotis, Georgios Paliouras, and Ioannis Kakadiaris. 2018. Results of the sixth edition of the BioASQ challenge. In *Proceedings of the 6th BioASQ Workshop A challenge on large-scale biomedical semantic indexing and question answering*, pages 1–10, Brussels, Belgium. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad.

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

Abhilasha Ravichander, Alan W Black, Shomir Wilson, Thomas Norton, and Norman Sadeh. 2019. Question answering for privacy policies: Combining computational and legal perspectives. 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 4947–4958, Hong Kong, China. Association for Computational Linguistics.

Marzieh Saeidi, Max Bartolo, Patrick Lewis, Sameer Singh, Tim Rocktäschel, Mike Sheldon, Guillaume Bouchard, and Sebastian Riedel. 2018. Interpretation of natural language rules in conversational machine reading.

Haitian Sun, William W. Cohen, and Ruslan Salakhutdinov. 2021. End-to-end multihop retrieval for compositional question answering over long documents.

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

Pat Verga, Haitian Sun, Livio Baldini Soares, and William W. Cohen. 2020. Facts as experts: Adaptable and interpretable neural memory over symbolic knowledge.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents.

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

Michael J. Q. Zhang and Eunsol Choi. 2021. Situatedqa: Incorporating extra-linguistic contexts into qa.

	Type	#
Answer type	yes / no	1751
	extractive	1527
Condition type	deterministic	2475
	conditional	803
Number of answers	single	2526
	multiple	752
–	not answerable	149

Table 5: Statistics on different types of questions.

A Dataset Analysis

The dataset consists of yes/no questions and extractive questions. Questions may contain one or more answers, with or without conditions. The statistics of the questions are shown in Table 5.

Answer type Among all the answerable questions, 1751 questions have yes/no answers while the other 1527 questions have extractive answers. 1161 of the yes/no questions have the answer "yes", 712 questions have answer "no", and 122 questions have both answers "yes" and "no" under different conditions. Please see the example in Table 1. The average length of the extract answers is 6.36 tokens.

Condition type 803 questions have conditional answers. 390 out of the 803 questions have one answer, but this answer is only correct if the conditions are satisfied. 173 questions have multiple answers, each have their own conditions, i.e. the answers are different if different conditions apply. The rest 240 questions also have multiple answers, but some of the answers require conditions while other don't. See examples in Table 1. A total of 1090 answers from 803 questions have conditions, among which 672 answers have only one condition and 418 answers have multiple conditions.

Number of answers Besides questions that have different answers under different conditions, 339 questions have multiple deterministic answers.