Chain of Condition: Construct, Verify and Solve Conditions for Conditional Question Answering

Anonymous ACL submission

Abstract

Conditional question answering (CQA) is an important task that aims to find probable answers and identify conditions that need to be satisfied to support the answer. Existing approaches struggle with CQA due to two main challenges: (1) precisely identifying conditions and their logical relationship, and (2) verifying and solving the conditions. To address these challenges, we propose Chain of Condition, a novel prompting approach by firstly identifying all conditions and constructing their logical relationships explicitly according to the document, then verifying whether these conditions are satisfied, finally solving the logical expression by tools to indicate any missing conditions and generating the answer based on the resolved conditions. The experiments on two benchmark conditional question answering datasets shows chain of condition outperforms existing prompting baselines, establishing a new state-of-the-art. Furthermore, with backbone models like GPT-3.5-Turbo or GPT-4, it surpasses all supervised baselines with only few-shot settings. [1]

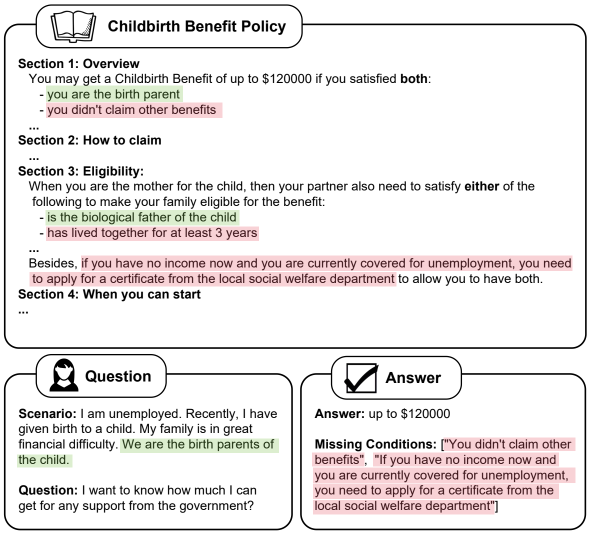

Figure 1: An example of conditional question answering. All conditions are colored. The conditions in green are satisfied by the user's description, while those in red are not mentioned. The second red condition, *has lived together for at least 3 years*, is not necessary because it has an "either" relationship with an already satisfied condition. But the other two red conditions are required to make the answer "up to $120000" correct.

1 Introduction

Conditional question answering (CQA) aims to answer questions where the information provided by the user may not be sufficient, therefore, additional conditions are necessary to imply the correctness of the answers (Saeidi et al., 2018; Min et al., 2020; Sun et al., 2021a; Dhingra et al., 2022). CQA is a challenging and promising task, which has been gaining increasing attention recently (Sun et al., 2022; Du et al., 2023; Wang et al., 2023; Hussain et al., 2023; Puerto et al., 2024). An example is shown in Figure 1. The user asks for the amount of benefit she would receive, but according to the policy, the applicant must not claim for other benefits and has an unemployment certificate as prerequisite. These conditions are not mentioned in the user's description, so a proper answer should include the amount of benefit as well as these unsatisfied conditions to ensure the eligibility.

The major challenge for CQA is twofold. The first challenge is identifying conditions from the document and comprehending the relationships between them. In Figure 1, all conditions are highlighted, while the rest of the description concerning other aspects of Childbirth Benefit is irrelevant to the question. Moreover, there are different relationships between these conditions. For example, the conditions *you are the birth parent* and *you didn't claim other benefits* must **both** be satisfied, whereas the conditions *is the biological father of the child* and *has lived together for at least 3 years* require **at least one** to be satisfied. Precisely identifying all conditions and understanding their relationships

[1] Codes will be released upon the acceptance of this paper.

according to the document is a fundamental prerequisite for the CQA task, but existing approaches primarily build end-to-end systems that overlook this challenge (Ainslie et al., 2020; Izacard and Grave, 2021; Hussain et al., 2023). These methods take the whole document as input, train models to implicitly identify conditions and parse their relationships, and directly output the answer along with any missing conditions. Consequently, due to the limitations of implicit reasoning capabilities in models, these approaches struggle with questions involving multiple conditions and complex relationships. Besides, their solution path is impossible for users to interpret.

Verifying and solving conditions with their logical relationships based on the user's question is the second challenge (Sun et al., 2022). Each condition may be satisfied, contradicted or ignored by the user, and conditions form a logical expression with each other based on their relationships. Solving this expression is necessary for accurately identifying all conditions missing from the user's input. For example, in Figure 1, the conditions in green are satisfied by the user, while those in red are not mentioned. Moreover, although the condition *has lived together for at least 3 years* is not mentioned by the user, it has an "either" relationship with an already satisfied condition, *is the biological father of the child*. Therefore, the condition *has lived together for at least 3 years* is not necessary for user to satisfy. Determining the unnecessity of this condition requires correctly solving the logical expression of conditions. Previous works train models to verify and solve conditions simultaneously, requiring models to implicitly resolve the expression (Du et al., 2023; Wang et al., 2023). This approach risks computational errors in solving expressions, and reduces the precision of predicting conditions.

To address these challenges, we introduce **Chain of Condition**, a novel prompting framework for constructing, verifying, and solving conditions in the CQA task. Chain of condition include three main steps: first explicitly identifying all conditions from the document and constructing the logical expression of them according to the document, next verifying whether conditions have been satisfied by the user, finally solving the logical expression precisely by tools to indicate any missing conditions, and generating the appropriate answer based on the resolved conditions.

We conduct experiments on two CQA benchmark datasets ConditionalQA (Sun et al., 2021a) and ShARC (Saeidi et al., 2018). The results show that chain of condition remarkably outperforms all prompting baselines. And with backbone models like GPT-3.5-Turbo or GPT-4, chain of condition even performs better than all supervised baselines.

Our contributions are summarized as follows:

(1)We are the first to investigate prompting LLMs for conditional question answering task.

(2)We propose Chain of Condition, a construct, verify and solve prompting framework. Experiments show chain of condition outperforms existing prompting baselines on all benchmark datasets, establishing a new state-of-the-art. And with backbone models like GPT-3.5-Turbo or GPT-4, it even surpasses all fully supervised baselines with only few-shot settings.

(3)We decompose the CQA task for LLMs, creating a coherent and interpretable reasoning path that is easier for humans to understand.

2 Related Works

Prompting Methods for LLMs Large language models can be guided to solve tasks in a step-by-step manner (Wei et al., 2023). For more complex reasoning tasks such as multi-hop QA (Yang et al., 2018) or math problems (Cobbe et al., 2021), previous works typically address them by decomposing the question into simple sub-questions for models to solve sequentially (Yao et al., 2023; Press et al., 2023; Servantez et al., 2024; Tao et al., 2024). This decomposition reduces task difficulty and improves interpretability. Another approach to enhance performance on reasoning tasks is by combining LLMs with symbolic interpreters such as a Python runtime (Chen et al., 2023; Gao et al., 2023; Lyu et al., 2023) or a SAT solver (Ye et al., 2023). Compared to textual reasoning, program-based reasoning is executed accurately, thus achieving high-precision reasoning in complex questions. Furthermore, Puerto et al. (2024) compared the performance of these two approaches and showed that introducing code in the prompt may elicit the reasoning ability for the CQA task. In this work, we integrate the ideas of decomposing task and leveraging code interpreter into our chain of condition framework, enabling it to benefit from both approaches' advantages. This not only improves interpretability but also increases precision.

Supervised Methods for the CQA task Various pre-trained models have been proposed for the con-

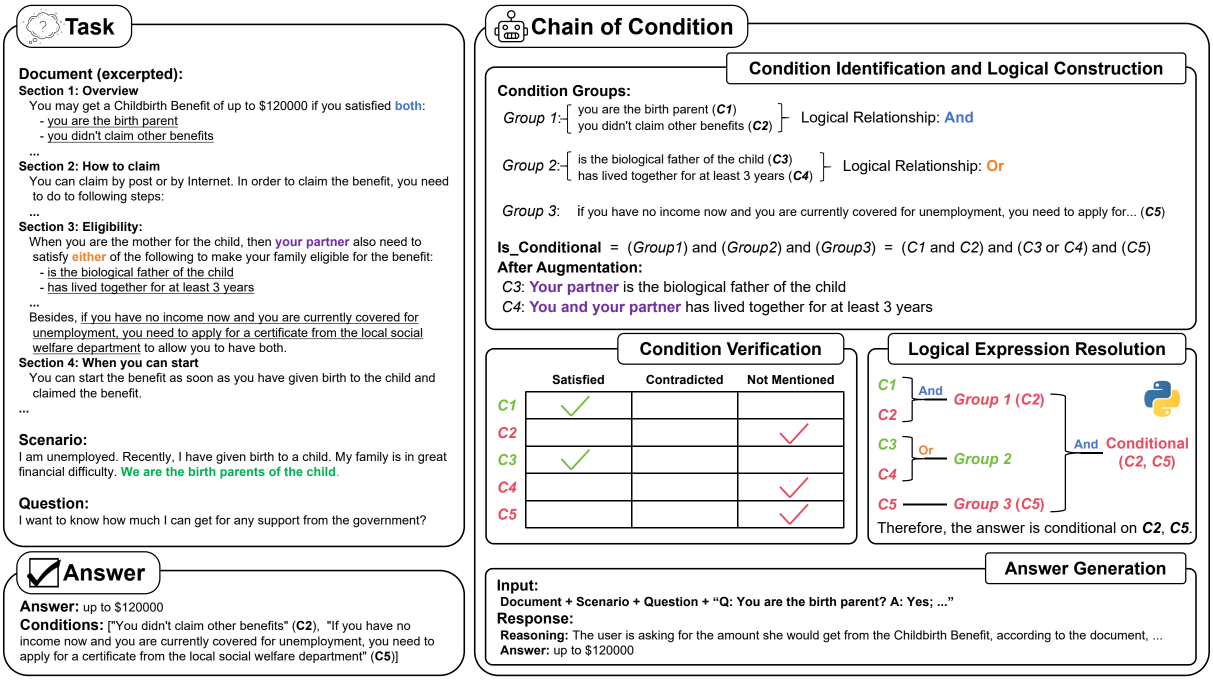

Figure 2: Method overview. Chain of Condition consists of three main steps: condition identification and logical construction, condition verification, expression solution and answer generation.

ditional question answering (CQA) task, including both extractive and generative models. Extractive models such as the ETC pipeline (Ainslie et al., 2020) and DocHopper (Sun et al., 2021b) extract answers and conditions from input content. Generative models, including FiD (Izacard and Grave, 2021), SDHG (Du et al., 2023), and TReasoner (Sun et al., 2022), leverage generative models to directly generate answers and conditions together. Additionally, Hussain et al. (2023) explored directly fine-tuning LLMs for the CQA task, demonstrating better performance but at a much higher training cost, while Wang et al. (2023) proposed the LSD framework to generate more conditional questions for fine-tuning. However, these methods are often limited to specific downstream fine-tuning tasks and lack generalizability. In contrast, chain of condition does not require further fine-tuning and exhibits better generalizability due to its few-shot setting.

3 Preliminary

We investigate the conditional question answering (CQA) task, where the answer is valid only when all missing conditions from the context are provided. Formally, the task's input consists of the user's **question** Q and **scenario** S paired with a reference **document** D. The answer should be inferred from the document. Unlike other QA tasks, the document in the CQA task contains numerous **conditions** $C = \{c_1; c_2; ...; c_n\} \subset D$ that must be satisfied to obtain the answer. The complete output includes the answer along with any corresponding unmentioned conditions $A = (a, C^{(u)})$, where $\{c_1^{(u)}; ...; c_k^{(u)}\} \subset C$ denotes the i-th unmentioned condition for answer a, and $k_i \geq 0$ denotes the total number of unmentioned conditions for the answer[2]. If there are no unmentioned conditions, then we categorize the answer as *deterministic*. Otherwise, we call it *conditional*, and all missing conditions should be listed simultaneously with the answer.

4 Methodology

We introduce **Chain of Condition**, a novel approach to guide Large Language Models (LLMs) for the conditional question answering task. Chain of condition pipeline is similar to the human process for dealing with such problems, and therefore demonstrates a more coherent and interpretable solving path for users to understand. The detailed process of this approach is illustrated in Figure 2.

Chain of condition includes three steps: condition identification and logical construction, condition verification, expression solution and answer generation. These steps decompose the original

[2] A few questions in ConditionalQA have multiple answers with conditions, and we leave the condition prediction for them as future work.

CQA task into smaller sub-tasks, allowing the model to solve them sequentially. First, we identify all conditions and parse their logical relationships according to the document, forming a logical expression of conditions. Secondly, we verify each condition's fulfillment in the user's description. Finally, We take the verification results to solve the logical expression and identify all missing conditions by tool, and generate the answer with the aid of condition solutions. A prompt example for each step is in Appendix F.

4.1 Condition Identification and Logical Construction

The document D contains a substantial amount of irrelevant information, so the first task is to filter this out and identify all relevant conditions $C = \{c_1; c_2; ...; c_n\} \subset D$. By leveraging the powerful language comprehension capabilities of LLMs, we can instruct the model to directly identify the locations of conditions in the document.

In addition to identifying the conditions, it is crucial to arrange them in a particular logical order according to the document. We address this by instructing models to construct condition groups $G = \{G_1; G_2; ...; G_k\}$, where k is the total number of condition groups in the document. The i-th condition group G_i is defined as $G_i(c_1^i, c_2^i, ..., c_{n_i}^i, o_i) = (f(c_1^i)\ o_i\ f(c_2^i)\ o_i\ ...\ o_i\ f(c_{n_i}^i))$, where $\{c_1^i, ..., c_{n_i}^i\}$ are the conditions in the i-th group, $o_i \in \{\text{and}, \text{or}\}$ is the logical operator connecting them and $f(c)$ represents the fulfillment of condition $c \in C$ that will be determined in the next step. We parse the model's output to obtain G, ultimately forming a compositional logical expression of conditions $F(G_1, ..., G_k, o) = ((G_1)\ o\ ...\ o\ (G_k))$, where $o \in \{\text{and}, \text{or}\}$. The solving result of G_i and F are in $\{\bar{\mathbf{d}}, \bar{\mathbf{c}}\}$, where $\bar{\mathbf{d}}$ denote the *deterministic* answer and $\bar{\mathbf{c}}$ *conditional* answer.

Besides, the conditions in the document are usually free-form, making it difficult to thoroughly separate a condition from other irrelevant context. Simply truncating or slicing the document may result in incomplete conditions. For example, in Figure 2, the condition *is the biological father of the child* lacks a subject, which needs to be extracted from the previous sentence, *your partner*. This incompleteness could prevent the model from correctly understanding the meaning of the condition, consequently hindering its ability to accurately verify the fulfillment of the condition.

To address this problem, we take context-based augmentation after identifying condition's locations. This approach allows us to obtain a short paragraph for each condition, containing all the necessary additional information. Specifically, we employ two augmentation methods: leveraging the structural information of the document or using transcription. When the document has a certain structure, such as HTML tags for each paragraph, we use this to find the relevant context for augmentation. We take the entire subsection where the condition appears as the augmentation paragraph, ensuring it contains enough background information while being much shorter than the entire document. When there is no structural information available, we instruct the model to directly transcribe the condition based on the context.

4.2 Condition Verification

After acquiring all condition groups and augmenting each condition, we instruct the models to verify the fulfillment of each condition sequentially. This involves taking the question, scenario, and augmented condition as input, and leveraging the powerful reasoning capabilities of LLMs to determine the status of each condition. For each condition $c \in C$, the verification process can be formalized as determining the value of function $f(c) \in \{\tilde{\mathbf{s}}, \tilde{\mathbf{c}}, \tilde{\mathbf{n}}\}$. Here, $\tilde{\mathbf{s}}$ means the condition is satisfied by the user, $\tilde{\mathbf{c}}$ means it is contradicted, and $\tilde{\mathbf{n}}$ means the condition is not mentioned. Conditions that are either satisfied or contradicted lead to a *deterministic* answer, while conditions that are not mentioned result in a *conditional* answer. Therefore, the solving process of the expression F can be seen as boolean operations on true/false values $f(c)$.

4.3 Expression Solution and Answer Generation

After obtaining the verification result $f(c)$ for each condition c, we need to recompose these results into groups G and logical expression F, and follow the document to determine whether the answer is *conditional* or *deterministic*. And if the answer is *conditional*, all missing conditions should be listed along with the answer.

Traditionally, this is done by prompting models to implicitly reason and resolve the logical expression. However, recent studies have shown that even large language models struggle with logical or mathematical reasoning tasks (Blair-Stanek et al., 2023). Therefore, a better solution is to offload the computation process to an external symbolic interpreter (Chen et al., 2023; Gao et al., 2023; Lyu et al., 2023; Ye et al., 2023). In chain of condition, we use a Python interpreter to solve the logical expression $F(G_1, ..., G_k, o)$. For *conditional* answers, we also identify all missing conditions $C^{(\mathrm{u})}$ by $\{c_i^j | f(c_i^j) = \tilde{\mathbf{n}}\ ,\ G_i = \bar{\mathbf{c}}, \forall i \leq n_i, \forall j \leq k\}$. This approach reduces model inference costs, improves precision, and enhances interpretability.

After obtaining the complete result for the conditions, we instruct the models to generate the answer. Since we have already verified each condition's fulfillment, we can leverage this information for more accurate answer generation. Specifically, we add these conditions c along with their fulfillment $f(c)$ into the prompt. This provides the model with straightforward information about the conditions, reducing the need to repeatedly infer their fulfillment from the document. Additionally, these conditions help the model locate relevant paragraphs about the question in the document.

5 Experimental Setup

5.1 Datasets and Evaluation Metrics

Throughout our experiments, we use two conditional question answering datasets: ConditionalQA (Sun et al., 2021a) and ShARC (Saeidi et al., 2018). More information about these datasets are in Appendix A.

ConditionalQA is a dataset features at long and complex documents, and has many different types of question. The document in ConditionalQA is well-structured, because it is directly crawled from websites and contains HTML tags for each paragraph. This brings the convenience for condition identification and augmentation.

We use the metrics from the original paper (Sun et al., 2021a) for evaluation, which includes two sets of metrics: EM/F1 and conditional EM/F1 (abbreviated as **w/conds**). EM measures the exact match of predicted answer spans with gold ones, while F1 is the harmonic mean of token-level precision and recall. Conditional EM/F1 jointly measures the correctness of answer spans and the predicted conditions, providing a more comprehensive assessment of a model's performance on the CQA task. The exact metric computation functions are in Appendix C.

ShARC is a conversational QA dataset, and the original task is to answer the question if the information in the dialog history is enough, or to generate a new question to acquire missing information. We follow the previous work (Puerto et al., 2024) to isolate the QA task from the conversational setting to form a benchmark of the CQA task, resulting in a dataset that the model only needs to answer "yes", "no" or "not enough information". Additionally, we discard all irrelevant questions from the dataset for better measurement.

We evaluating model's accuracy on the ternary classification about the answer. Since there are no human annotated conditions in the dataset, so it is not possible to further measure the accuracy of missing conditions predicted by the model, and we leave the more precise evaluation of ShARC as future work.

5.2 Baselines

Prompting Baselines We compare our approach, chain of condition, with 4 different prompting baselines in total.

• **Code Prompting** (Puerto et al., 2024) is the only existing approach for prompting models for the CQA task as far as we know. This method extend the original text prompt with additional LLM-generated codes, which elicits the model's conditional reasoning abilities for CQA tasks.

• **Self-Ask** (Press et al., 2023) is a recently proposed, well-performing prompting method, and we adapt it for the CQA task. This method decompose the question by explicitly asking and answering intermediate questions until reaching the final answer and missing conditions.

Additionally, we use **Zero Shot** prompting and **Chain of Thought** prompting (Wei et al., 2023) as our baselines.

Supervised Baselines The supervised baselines for ConditionalQA include **ETC pipeline** (Ainslie et al., 2020), **DocHopper** (Sun et al., 2021b), **FID** (Izacard and Grave, 2021), **SDHG** (Du et al., 2023), **TReasoner** (Sun et al., 2022), and **LSD** (Wang et al., 2023).

For ShARC, since we follow previous work to modify the dataset's output format and discard all irrelevant instances(Puerto et al., 2024), no available supervised baselines exist. Therefore, we only compare chain of condition with other prompting baselines mentioned above.

5.3 LLM Setup

We conduct our experiments on four different large language models (LLMs) to investigate whether chain of condition performs consistently better

Method	GPT-3.5		Llama-2 (70B)		Llama-2 (13B)		Mistral		Average	
	EM/F1	w/conds	EM/F1	w/conds	EM/F1	w/conds	EM/F1	w/conds	EM/F1	w/conds
Zero-Shot	59.5/71.0	23.9/29.5	44.0/51.2	26.6/30.9	42.3/49.6	26.1/28.9	44.2/50.7	26.7/30.8	47.5/55.6	25.8/30.0
Chain of Thought	59.3/70.0	45.4/54.6	62.2/71.4	45.5/53.7	56.8/65.8	38.7/44.8	**58.3/68.6**	37.7/46.4	59.2/69.0	41.8/49.9
Code Prompting	60.4/68.2	50.8/57.5	54.4/63.1	15.9/19.2	45.9/49.7	11.0/12.3	48.4/52.3	10.4/10.6	52.3/58.3	22.0/24.9
Self-Ask	54.9/66.9	41.3/52.2	59.2/69.9	36.1/45.5	47.9/59.9	30.3/38.3	49.6/60.5	**41.2/50.4**	52.9/64.3	37.2/46.6
Chain of Condition	**64.6/73.7**	**52.9/61.0**	**64.7/75.2**	**47.7/56.0**	**57.2/67.1**	**43.0/51.3**	55.5/63.8	40.7/47.5	**60.5/70.0**	**46.1/54.0**

Table 1: Result of prompting methods on ConditionalQA. The best scores are made **bold**, with the second underlined.

Method	GPT-3.5	Llama-2 (70B)	Llama-2 (13B)	Mistral	Average
Zero-Shot	63.2	43.8	45.5	36.9	47.4
Chain of Thought	66.7	69.6	63.0	60.2	64.9
Code Prompting	60.4	39.9	37.6	40.3	44.6
Self-Ask	**70.3**	69.1	**67.4**	60.5	66.8
Chain of Condition	70.2	**74.9**	64.2	**61.8**	**67.8**

Table 2: Result of prompting methods on ShARC. The best scores are made **bold**, with the second underlined.

Method	EM/F1	w/conds
Supervised Baselines		
SDHG	49.0/56.5	39.0/46.0
TReasoner	57.2/63.5	46.1/51.9
LSD+Longformer	58.7/66.2	45.0/50.5
Chain of Condition		
GPT-3.5 (Retrieval)	56.6/66.2	42.1/51.0
GPT-3.5 (16K)	**61.0/70.0**	**48.5/56.0**
GPT-3.5 (Oracle)	64.6/73.7	52.9/61.0
GPT-4 (Oracle)	70.8/79.5	56.9/63.0

Table 3: Results of ConditionalQA compared with supervised baselines.

across various settings. We use a commercial model, GPT-3.5-Turbo, and three open-source models, Llama-2-70B-chat, Llama-2-13B-chat, and Mistral-7B. Additionally, we leverage GPT-4 (OpenAI, 2023) for limited experiments exclusively on ConditionalQA due to cost constraints. For all models, we set the temperature to 0.0 to ensure reproducibility of the results, while using default settings for others.

The original documents in ConditionalQA can be up to 9320 tokens long, exceeding the context limitations of many LLMs, posing a challenge for all prompting methods. This could be solved by introducing a retriever to retrieve only relevant paragraphs of the document. Therefore, to address this issue and eliminate the interference from retriever performance in our experiments, we use an oracle retriever to select relevant passages for the question. We follow the methodology of previous work (Puerto et al., 2024) by retaining all sections that include at least one human-annotated gold evidence and concatenating them to form the input.

And for comparison with supervised methods, we employ two approaches: (1) Using a retriever to retrieve relevant paragraphs from the document, and (2) Using a long-context version of an LLM as our backbone model. The results of these approaches will be discussed in Section 6.1.

See more setup details in Appendix B.

6 Results and Analysis

6.1 Main Results

We report the performance of chain of condition and all baselines on two benchmark datasets[3]. Table 1 presents the performance of all prompting methods on ConditionalQA, while Table 2 shows the performance on ShARC. Table 3 compares the results of chain of condition with all supervised baselines on ConditionalQA.

The original evaluation script of ConditionalQA provides not only the overall result but also a detailed breakdown by question type in the dev set. We report the overall results here, with more detailed results available in Appendix E.

Chain of condition outperforms all prompting baselines on each dataset. It surpasses all other baselines on both datasets, establishing a new state-of-the-art. Additionally, Self-Ask also performs relatively well on ShARC, which can be attributed to the dataset's features. The conversational format of ShARC is naturally suitable for leveraging Self-Ask, making it reasonable for it to perform better

[3]The test set of ConditionalQA is not publicly available, and the question number is larger than dev set, causing a much higher api cost. Thus we only evaluate all methods on dev set.

than Zero-Shot or Chain of Thought prompting.

Chain of condition outperforms all supervised baselines. With backbone models like GPT-3.5-Turbo or GPT-4, it surpasses all supervised baselines with few-shot settings. This result highlights the promising future of prompting methods for the CQA task, not only achieving better performance but also reducing the costs for fine-tuning.

Moreover, the performance of prompting GPT-3.5 with retrieved passages is much lower than prompting GPT-3.5 with gold evidence or prompting GPT-3.5-16k with the full document. This indicates that the performance of the retriever is a bottleneck that limits the effectiveness of chain of condition. It also suggests the substantial potential of it when augmented with a better retriever.

6.2 Analysis

In this section, we first conduct ablation studies with GPT-3.5 on the chain of condition framework to demonstrate the necessity of each step. Next, we show that chain of condition consistently outperforms all baselines in more challenging task settings, and finally analyze the reasons for its superior performance.

Explicitly identifying and constructing conditions is crucial. The importance of this step lies in two aspects. First, it ensures the model identifies all possible conditions and can explicitly solve the logical expressions by external tools, improving performance on conditions. Secondly, it allows us to generate condition-aided answer, which is only feasible if all conditions are explicitly identified.

To prove the first hypothese, we conduct an ablation study on ConditionalQA because it has gold-labeled missing conditions. In this study, we prompted GPT-3.5 to first identify all conditions, then check their fulfillment, and finally indicate all unmentioned conditions implicitly through reasoning. As shown in Table 4, this ablation results in a drop of 3.2 EM score and 2.8 F1 score for answer measurement, as well as 17.4 EM score and 19.5 F1 score for joint answer and condition measurement. The performance drop is much greater when measuring both the answer and condition compared to measuring the answer alone, which indicates that removing the condition identification step leads to a much larger decrease in accuracy for conditions. Further investigation into the model's output reveals that the average number of predicted missing conditions for *conditional* answers by the model increases from 1.27 to 1.67, suggesting that the model tends to judge conditions as not mentioned by the user more frequently when the condition identification step is omitted.

	EM	F1	Cond EM	Cond F1
Chain of Condition	64.6	73.7	52.9	61.0
Prompting Only	61.4	70.9	35.9	42.9
Δ	*-3.2*	*-2.8*	*-17.0*	*-18.1*
And Only	62.2	71.0	48.9	56.3
Δ	*-2.4*	*-2.7*	*-4.0*	*-4.7*
Or Only	61.2	69.9	40.1	45.6
Δ	*-3.4*	*-3.8*	*-12.8*	*-15.4*

Table 4: Ablation study for explicitly identifying and constructing conditions on ConditionalQA.

	ConditionalQA EM/F1	ConditionalQA w/conds	ShARC Accuracy
Chain of Condition	64.6/73.7	52.9/61.0	70.2
w/o Results	61.4/70.9	50.5/59.1	67.5
Δ	*-3.2/-2.8*	*-2.4/-1.9*	*-2.7*

Table 5: Ablation study for answer generation on ConditionalQA and ShARC. *w/o Results* refers to removing condition verification results from the answer generation input.

We also conduct an ablation for the necessity of using both logical operators "and" and "or". We remove each of them and prompt the model using chain of condition. The results in Table 4 indicate removing either operator reduces the performance.

The discussion of the second hypothesis is covered in the ablation study of the answer generation step in the following paragraphs.

Condition augmentation improves verification accuracy. The removal of contextual information can hinder the model's ability to correctly understand the meaning of a condition. To prove this, we leverage the structured document of ConditionalQA to conduct an ablation study. We remove all other paragraphs of the condition's subsection, keeping only the original condition as input for verification. The result of conditional EM drops by 2.4 from 52.9 to 50.5, and the conditional F1 drops by 2.5 from 61.0 to 58.5 for this setting on GPT-3.5, indicating that the performance of condition prediction decreases due to reduced verification accuracy. Thus, condition augmentation would improve verification precision.

Including verification results helps answer generation. In this ablation, we remove the verification results of conditions from the input of answer generation. The results are shown in Table 5. The performance drops by 3.2 EM score and 2.8 F1 score for answers, and by 2.4 EM score and 1.9 F1 score when jointly measuring answers and conditions on ConditionalQA. Additionally, the accuracy drops by 2.7 on ShARC.

Furthermore, we find that the performance drop on ConditionalQA is mostly attributed to the yes/no type questions, with a drop of 7.0 EM/F1 score and 5.3 conditional EM/F1 score. The likely reason for this phenomenon lies in the answer determination procedure: an extracted span-type answer can be found directly in the document even without verifying any condition. However, a yes/no answer must be inferred from the document along with each condition's fulfillment. Therefore, including the conditions' fulfillment in the prompt helps the model by reducing the need to repeatedly infer their fulfillment, allowing it to directly synthesize the information to generate the final answer.

Chain of condition performs better at indicating missing conditions. Most questions in these CQA datasets involve identifying and solving conditions, but only a small portion of them are truly *conditional*. This is because, in many cases, the conditions for the answer are all satisfied by the user's scenario, so the model only needs to give a correct judgement on whether the answer is *conditional*. However, when we consider only the *conditional* answers in the dataset, correctly addressing them becomes more challenging. This is because the model not only needs to properly generate the answer and determine it is *conditional*, but also precisely indicate the missing conditions.

The experimental results support this intuition. The performance of all methods on ConditionalQA[4] greatly drops when considering only the *conditional* answers, as shown in Table 6. Besides, chain of condition consistently outperforms other prompting baselines in this setting, demonstrating its effectiveness in indicating missing conditions.

Furthermore, in order to analyze the reasons behind chain of condition's superior performance on the CQA task, we divide the dev set of ConditionalQA based on the total number of gold conditions for each question in the document, resulting in two question groups. The first group contains data with at most one conditions, while the second group has at least three conditions, indicating a more complex set of conditions for solving.

[4]ShARC does not have human-annotated conditions, so we could not experiment on it.

Conditional	EM	F1	Cond EM	Cond F1
Zero-Shot	40.7	49.1	12.9	16.0
CoT	45.8	53.6	13.1	16.4
Code	47.2	54.3	8.5	11.5
Self-Ask	49.7	58.3	13.5	17.5
Ours	**56.0**	**62.2**	**18.9**	**20.7**

Table 6: Result of different prompting methods on *conditional* answer questions.

#Conds	<=1		>=3	
Groups	EM/F1	C_F1	EM/F1	C_F1
Zero-Shot	64.0/76.1	53.8	47.7/50.7	8.9
CoT	60.5/71.6	77.3	44.0/47.7	16.5
Code	61.9/70.1	**87.9**	51.7/57.0	4.0
Self-Ask	55.0/68.1	77.6	54.0/59.1	20.1
Ours	**65.5/75.3**	84.5	**60.0/65.2**	**31.6**

Table 7: Performance on 2 groups in ConditionalQA. *C_F1* is the F1 score of predicted conditions.

We report the performance of GPT-3.5 with all prompting methods on these two groups in Table 7. Since there isn't a metric that directly measures the quality of predicted conditions, we additionally report the F1 score of the predicted conditions. The results highlight the increased difficulty of questions involving complex conditions, and chain of condition shows much less performance degradation in this more complex group. This indicates its superior ability to handle complex conditions. We attribute this to the explicit identification of conditions and the use of a code interpreter to resolve the logical relationships between conditions.

7 Conclusion

In this work, we propose Chain of Condition, a novel prompting approach for conditional question answering. It prompts models to identify conditions with logical expressions and introduces a Python interpreter for resolution, effectively improving precision and enhancing interpretability. We conduct experiments demonstrating that chain of condition outperforms existing prompting baselines on all benchmark datasets. Additionally, when utilizing a strong backbone model, it surpasses supervised baselines. Our work analyzes the challenges associated with CQA and highlights the importance of condition identification, paving the way for future research directions.

Limitations

While chain of condition outperforms all baselines, it faces limitations in situations where document length exceeds the LLM's input context capacity, primarily due to the retriever's poor performance. This shortcoming can reduce the effectiveness of our approach in more realistic scenarios. Additionally, the multi-step prompting framework in chain of condition decomposes the original CQA task into numerous sub-tasks, leading to lower token efficiency compared to simpler prompting baselines and typically requiring more tokens to solve the entire problem. Another limitation is the scarcity of CQA datasets, which hampers further research in this area.

References

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. *Preprint*, arxiv:2004.08483.

Andrew Blair-Stanek, Nils Holzenberger, and Benjamin Van Durme. 2023. Can gpt-3 perform statutory reasoning? *Preprint*, arXiv:2302.06100.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Preprint*, arXiv:2211.12588.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Bhuwan Dhingra, Jeremy R. Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W. Cohen. 2022. Time-aware language models as temporal knowledge bases. *Transactions of the Association for Computational Linguistics*, 10:257–273.

Haowei Du, Yansong Feng, Chen Li, Yang Li, Yunshi Lan, and Dongyan Zhao. 2023. Structure-discourse hierarchical graph for conditional question answering on long documents. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6282–6293, Toronto, Canada. Association for Computational Linguistics.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. *Preprint*, arXiv:2211.10435.

Syed-Amad Hussain, Parag Pravin Dakle, SaiKrishna Rallabandi, and Preethi Raghavan. 2023. Towards leveraging LLMs for Conditional QA. *Preprint*, arxiv:2312.01143.

Gautier Izacard and Edouard Grave. 2021. (FiD)Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online. Association for Computational Linguistics.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 305–329, Nusa Dua, Bali. Association for Computational Linguistics.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. AmbigQA: Answering ambiguous open-domain questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5783–5797, Online. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore. Association for Computational Linguistics.

Haritz Puerto, Martin Tutek, Somak Aditya, Xiaodan Zhu, and Iryna Gurevych. 2024. Code Prompting Elicits Conditional Reasoning Abilities in Text+Code LLMs. *Preprint*, arxiv:2401.10065.

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. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2087–2097.

Sergio Servantez, Joe Barrow, Kristian Hammond, and Rajiv Jain. 2024. Chain of logic: Rule-based reasoning with large language models. *Preprint*, arXiv:2402.10400.

Haitian Sun, William W. Cohen, and Ruslan Salakhutdinov. 2021a. ConditionalQA: A Complex Reading Comprehension Dataset with Conditional Answers. *Preprint*, arxiv:2110.06884.

Haitian Sun, William W. Cohen, and Ruslan Salakhutdinov. 2021b. Iterative Hierarchical Attention for Answering Complex Questions over Long Documents. *Preprint*, arxiv:2106.00200.

Haitian Sun, William W. Cohen, and Ruslan Salakhutdinov. 2022. Reasoning over logically interacted conditions for question answering. *Preprint*, arXiv:2205.12898.

Mingxu Tao, Dongyan Zhao, and Yansong Feng. 2024. Chain-of-discussion: A multi-model framework for complex evidence-based question answering. *Preprint*, arXiv:2402.16313.

Zihan Wang, Hongjin Qian, and Zhicheng Dou. 2023. Learning on Structured Documents for Conditional Question Answering. In *Chinese Computational Linguistics*, volume 14232, pages 37–57, Singapore. Springer Nature Singapore.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

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. *Preprint*, arXiv:1809.09600.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. *Preprint*, arxiv:2210.03629.

Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. 2023. Satlm: Satisfiability-aided language models using declarative prompting. *Preprint*, arXiv:2305.09656.

A Datasets

We use two benchmark datasets for evaluation: ConditionalQA (Sun et al., 2021a) and ShARC (Saeidi et al., 2018). The distribution of different question types in these datasets is presented in Table 8, with additional details about the datasets provided in Table 9.

ConditionalQA is a challenging benchmark for conditional question answering. It comprises a total of 3,427 questions of varying types, including yes/no questions, free-form extractive questions, questions with multiple answers, and non-answerable questions. Additionally, ConditionalQA categorizes questions into two types: *deterministic*, where all necessary conditions are already satisfied within the question, and *conditional*, where the complete answer must include those unsatisfied conditions as well.

ShARC is a conversational QA dataset with natural language document that has conditions where questions may be underspecified, and follow-up questions are needed to finally reach the answer. And when the conditions are all satisfied, the answer could be either *yes* or *no*. There are some questions in it that are irrelevant to the conditions, and we discard them for simplicity. At the time we conducted our experiments, the test set was not yet publicly available, so we follow Puerto et al. (2024) to random divide the dev set into two equal partitions and use one for experiment.

B LLM Setup

The exact models we used are as follows: GPT-3.5-0613, GPT-3.5-16k-0613, GPT-4-1106-Preview, Llama-2-70B-chat, Llama-2-13B-chat, and Mistral 7B v0.1. We ran the GPT models through the Azure AI service, and the other models on Nvidia A800. We used text-embedding-ada-002 as our retriever when comparing performance with supervised methods.

For all experiments, we used a seed of 42. The number of demonstrations for the baselines were as follows: 4 for chain of thought prompting, 3 for code prompting, and 4 for self-ask. In our approach, chain of condition, we used 4 exemplars for condition identification and logical construction, 6 for condition verification, and 4 for answer generation.

	Type	Number
ConditionalQA		
Answer type	yes/no	1751
	extractive	1527
Condition type	deterministic	2475
	conditional	803
Answer number	single	2526
	multiple	752
not answerable		149
ShARC		
Answer type	yes/no	15400
	follow-up	6814
	irrelevant	1946

Table 8: Question type statistics.

Dataset	ConditionalQA	ShARC
Training	2338	21890
Dev	285	1135
Test	804	1135
License	BSD 2	CC-BY-SA-3.0

Table 9: Dataset details.

C ConditionalQA Evaluation Metrics

The evaluation metrics for ConditionQA include four key metrics: EM (exact match), F1, Conditional EM, and Conditional F1. EM and F1 are commonly used in QA tasks. Given a list of predicted answers $\{\hat{a}_1, ..., \hat{a}_m\}$ and a list of reference answers $\{a_1, ..., a_n\}$, these metrics are computed as follows:

$$
\begin{aligned}
\text{EM} &= \max_{\{\tilde{a}_1,\ldots,\tilde{a}_m\}} \frac{\sum_{i=1}^{min(m,n)} s_{em}(\tilde{a}_i, a_i) \cdot \gamma_{m,n}}{n} \\
\text{F1} &= \max_{\{\tilde{a}_1,\ldots,\tilde{a}_m\}} \frac{\sum_{i=1}^{min(m,n)} s_{f1}(\tilde{a}_i, a_i) \cdot \gamma_{m,n}}{n} \\
\gamma_{m,n} &= \begin{cases} e^{1-m/n} & \text{if } m > n \\ 1 & \text{if } m \leq n \end{cases}
\end{aligned}
\tag{817}
$$

Where $\{\tilde{a}_1, ..., \tilde{a}_m\}$ is a permutation of the predicted answers $\{\hat{a}_1, ..., \hat{a}_m\}$, s_{em} and s_{f1} are scoring functions that measures EM and F1 between two text spans. $\gamma_{m,n}$ is a penalty factor for the number of predicted answers.

While EM and F1 can evaluate the model's performance on answer prediction, they do not account for the accuracy of conditions associated with these answers. To jointly measure the performance of

	FP	IE	VE	Total
#Conds	57.8	55.0	17.5	130.3
Ratio	44.4%	42.2%	13.4%	100%

Table 10: Prediction Error on ConditionalQA.

both answers and conditions, Sun et al. (2021a) extended the scoring functions of EM and F1 to incorporate the prediction accuracy of conditions, resulting in two new metrics: Conditional EM and Conditional F1. These new scoring functions are computed as follows:

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

$$s_{f1+c}(\tilde{a}_i, \tilde{C}_i, a_i, C_i) = s_{f1}(\tilde{a}_i, a_i) \cdot \mathrm{F1}(\tilde{C}_i, C_i)$$

$$\mathrm{EM}_{+\mathrm{c}} = \max_{\{\tilde{a}_1,\ldots,\tilde{a}_m\}} \frac{\sum_{i=1}^{min(m,n)} s_{em+c}(\tilde{a}_i, \tilde{C}_i, a_i, C_i) \cdot \gamma_{m,n}}{n}$$

$$\mathrm{F1}_{+\mathrm{c}} = \max_{\{\tilde{a}_1,\ldots,\tilde{a}_m\}} \frac{\sum_{i=1}^{min(m,n)} s_{f1+c}(\tilde{a}_i, \tilde{C}_i, a_i, C_i) \cdot \gamma_{m,n}}{n}$$

Here, $\tilde{C}_i$ represents the set of conditions predicted by the model corresponding to the answer $\tilde{a}_i$, and C_i represents the oracle (ground truth) set of conditions. $\mathrm{F1}(\tilde{C}_i, C_i)$ denotes the HTML element level F1 score between the predicted set of conditions and the oracle set of conditions.

D Error Analysis

We investigate the prediction errors in ConditionalQA. We report detailed statistics for condition prediction. Errors are classified into False Positive (FP) and False Negative (FN) categories. Since chain of condition explicitly identifies all conditions, False Negatives can be further classified into Identifying Errors (IE) and Verification Errors (VE) based on the step at which the model makes mistakes. The results, averaged across four models, are shown in Table 10.

E More Detailed Results

We report the detailed results on ConditionalQA according to different question types in Table 11 for GPT-3.5, Table 12 for Llama-2 (70B), Table 13 for Llama-2 (13B) and Table 14 for Mistral.

F Prompt Examples

We provide an example of the prompt for condition identification and logical construction in Table 15 for ConditionalQA and in Table 18 for ShARC. We provide an example for condition verification in Table 16 for ConditionalQA and in Table 19 for ShARC. We provide an example for answer generation in Table 17 for ConditionalQA and in Table 20 for ShARC.

	Yes/No		Extractive		Conditional		Overall	
	EM/F1	**w/conds**	**EM/F1**	**w/conds**	**EM/F1**	**w/conds**	**EM/F1**	**w/conds**
Zero-Shot	82.1/82.1	17.0/17.0	29.5/55.0	19.7/32.1	40.7/49.1	12.9/16.0	59.5/71.0	23.9/29.5
Chain of Thought	80.4/80.4	54.2/54.2	31.3/55.0	29.6/**50.1**	45.8/53.6	13.1/16.4	59.3/70.0	45.4/54.6
Code Prompting	81.1/81.1	62.6/62.6	32.9/50.2	**32.2**/47.2	47.2/54.3	8.5/11.5	60.4/68.2	50.8/57.5
Self-Ask	76.2/76.2	49.8/49.8	26.1/52.8	25.3/49.7	49.7/58.3	13.5/17.5	54.9/66.9	41.3/52.2
Chain of Condition	**87.4/87.4**	**67.1/67.1**	**35.2/55.6**	31.8/50.0	**56.0/62.2**	**18.9/20.7**	**64.6/73.7**	**52.9/61.0**

Table 11: Result of different prompting methods on GPT-3.5-Turbo.

	Yes/No		Extractive		Conditional		Overall	
	EM/F1	**w/conds**	**EM/F1**	**w/conds**	**EM/F1**	**w/conds**	**EM/F1**	**w/conds**
Zero-Shot	68.2/68.2	36.9/36.9	10.8/27.0	7.1/16.7	52.9/55.7	9.2/9.5	44.0/51.2	26.6/30.9
Chain of Thought	78.2/78.2	49.1/49.1	**40.1/60.6**	**35.5/53.7**	48.7/53.8	11.4/13.6	62.2/71.4	45.5/53.7
Code Prompting	76.2/76.2	17.0/17.0	24.9/44.3	5.4/12.9	56.9/61.1	**19.6/21.2**	54.4/63.1	15.9/19.2
Self-Ask	79.7/79.7	35.4/35.4	31.8/55.6	30.0/50.9	53.1/61.6	17.6/20.9	59.2/69.9	36.1/45.5
Chain of Condition	**84.5/84.5**	**54.8/54.8**	35.4/60.4	32.2/52.0	**49.3/57.0**	17.4/19.6	**64.7/75.2**	**47.7/56.0**

Table 12: Result of different prompting methods on Llama-2(70B).

	Yes/No		Extractive		Conditional		Overall	
	EM/F1	**w/conds**	**EM/F1**	**w/conds**	**EM/F1**	**w/conds**	**EM/F1**	**w/conds**
Zero-Shot	66.4/66.4	35.7/35.7	9.1/25.3	6.3/14.4	**51.0/54.1**	6.2/7.2	42.3/49.6	26.1/28.9
Chain of Thought	69.7/69.7	40.5/40.5	**37.5/57.5**	**29.9**/43.6	42.0/50.2	10.6/13.5	56.8/65.8	38.7/44.8
Code Prompting	65.7/65.7	8.5/8.5	17.7/26.3	4.1/6.9	49.5/51.9	11.8/12.1	45.9/49.7	11.0/12.3
Self-Ask	65.7/65.7	34.4/34.4	22.2/48.9	18.1/36.0	35.2/41.5	5.7/7.5	47.9/59.9	30.3/38.3
Chain of Condition	**77.6/77.6**	**52.5/52.5**	29.8/51.6	26.2/**44.8**	45.8/52.6	**13.8/15.4**	**57.2/67.1**	**43.0/51.3**

Table 13: Result of different prompting methods on Llama-2(13B).

	Yes/No		Extractive		Conditional		Overall	
	EM/F1	**w/conds**	**EM/F1**	**w/conds**	**EM/F1**	**w/conds**	**EM/F1**	**w/conds**
Zero-Shot	68.5/68.5	36.3/36.3	10.8/25.4	7.9/17.0	**55.3**/56.9	4.7/5.1	44.1/50.7	26.7/30.8
Chain of Thought	**80.5/80.5**	41.2/41.2	**28.9/51.8**	**26.9/46.3**	51.4/**59.9**	15.9/**18.7**	**58.3/68.6**	37.7/46.4
Code Prompting	72.7/72.7	10.1/10.1	15.5/24.3	1.0/1.5	53.9/55.9	**16.9**/17.3	48.4/52.3	10.4/10.6
Self-Ask	74.1/74.1	**58.3/58.3**	16.7/40.8	15.6/36.2	41.9/50.7	13.7/17.0	49.6/60.5	**41.2/50.4**
Chain of Condition	80.4/80.4	51.9/51.9	22.8/41.3	21.7/36.8	43.4/48.1	9.0/11.1	55.5/63.8	40.7/47.5

Table 14: Result of different prompting methods on Mistral.

- Please find all the paragraphs that contain conditions for the suggested answer based on the provided documents.
- Provided Documents:
<h1>Joining a workplace pension</h1>
<p>All employers must provide a workplace pension scheme. This is called ‘automatic enrolment’.</p>
<p>Your employer must automatically enrol you into a pension scheme and make contributions to your pension if all of the following apply:</p>
you’re classed as a ‘worker’
you’re aged between 22 and State Pension age
you earn at least £10,000 per year
you usually (‘ordinarily’) work in the UK (read the detailed guidance if you’re not sure)
- Question: I am 23 years old. I have just got a new job in UK that will have a salary of £18,000 per year. I do not want to pay into a pension at this time. Does my employer need to automatically enrol me onto a pension?
- Suggested Answer: yes
- Response:
Reasoning: The user is asking whether the employer needs to automatically enrol the user onto a pension. According to the documents, the employer need automatically enrol the user into a pension scheme if the user is classed as a worker, aged between 22 and State Pension age, earns at least £10,000 per year, and usually works in the UK. So these are the prerequisites that all need to be satisfied, they are placed next to each other in the documents and are all needed to be satisfied, so they are in the same group and the relationship between them is "and".
Conditions: [{"contents": ["
you’re classed as a ‘worker’
", "
you’re aged between 22 and State Pension age
", "
you earn at least £10,000 per year
", "
you usually (‘ordinarily’) work in the UK (read the detailed guidance if you’re not sure)
"], "relationship": "and"}]

Table 15: ConditionalQA prompt example for condition identification and logical construction.

- Please check the relevance, find the condition and check the satisfaction of the condition. - Reference Document: Who must send a tax return   You must send a tax return if, in the last tax year (6 April to 5 April), you were: - self-employed as a ’sole trader’ and earned more than $31,000 (before taking off anything you can claim tax relief on)   - a partner in a business partnership   You will not usually need to send a return if your only income is from your wages or pension. But you may need to send one if you have any other untaxed income, such as: - money from renting out a property   - tips and commission   - Segment to Check: - self-employed as a ’sole trader’ and earned more than $31,000 (before taking off anything you can claim tax relief on)   - Question: I decided to start freelancing and working for myself and made freelancing my main job, taxes are not automatically deducted from my income. I have an income from my job for this tax year. How do I pay taxes on my income? - Suggested Answer: send a tax return - Response: Reasoning: The user is asking how to pay taxes on his income. The segment is about a prerequisite, and by looking at the reference document, we know this prerequisite is for the user to pay taxes return, so this segment is relevant to the question and answer, and contains a condition that the user needs to be self-employed as a "sole trader" and earned more than $31,000. Then let’s check the satisfaction of this in the question. We can see in the question, the user mentions that he decided to start freelancing and working for himself and made freelancing his main job, and he has an income for this tax year. But he does not explicitly mention the amount of income he earned, so we could not determine whether he earned more than $31,000, and the satisfaction of this condition is "not mentioned". Satisfaction: not mentioned

Table 16: ConditionalQA prompt example for condition verification.

- Please answer the question based on the provided documents and known information. - Provided Documents: Joining a workplace pension   All employers must provide a workplace pension scheme. This is called ‘automatic enrolment’.   Your employer must automatically enrol you into a pension scheme and make contributions to your pension if all of the following apply: - you’re aged between 22 and State Pension age   - you earn at least £10,000 per year   - you usually (‘ordinarily’) work in the UK (read the detailed guidance if you’re not sure)   - Question: I am 23 years old. I have just got a new job in UK that will have a salary of £8,000 per year. I do not want to pay into a pension at this time. Does my employer must enrol me onto a pension? - Known Information: Q: You’re aged between 22 and State Pension age? A: Yes; Q: You earn at least £10,000 per year? A: No; Q: You usually (‘ordinarily’) work in the UK (read the detailed guidance if you’re not sure)? A: Yes - Answer Type: yes/no - Response: Reasoning: The user is asking whether the employer needs to automatically enrol the user onto a pension. In the documents, the employer must automatically enrol the user into a pension scheme if the user is aged between 22 and State Pension age, and earns at least £10,000 per year, and usually works in the UK. In the known information, we know the user is aged between 22 and State Pension age, and usually works in the UK, but the user earns less than £10,000 per year. Therefore, the user does not satisfy the requirement of earning at least £10,000 per year, so the answer to the question is "no". Answer: No

Table 17: ConditionalQA prompt example for answer generation.

- Please find all conditions based on the provided documents.
- Provided Documents:

6. Immigration control

You may be able to get Child Tax Credit if you’re either working in the UK or you’re not working because you’re:

* retired
* pregnant or looking after children
* sick or disabled or your partner has died

- Question: I was born in North Carolina and my husband and I are still trying to conceive. May I be able to get Child Tax Credit?
- Response:

Reasoning: The user is asking whether she may be able to get Child Tax Credit. According to the documents, you may be able to get Child Tax Credit if you’re either working in the UK, or if you are not working because you’re retired, pregnant or looking after children, sick or disabled, or your partner has died. So there are precondition that the user needs to satisfy before she may be able to get Child Tax Credit, and there relationship is "or".

Conditions: {"contents": ["You’re working in the UK", "You’re not working because you’re retired", "You’re not working because you’re pregnant", "You’re not working because you’re looking after children", "You’re not working because you’re sick", "You’re not working because you’re disabled", "Your partner has died"], "relationship": "or"}

Table 18: ShARC prompt example for condition identification and logical construction.

- Please check the satisfaction of the given condition("satified" or "contradicted" or "not mentioned") based on the provided documents.
- Provided Documents:

6. Immigration control

You may be able to get Child Tax Credit if you’re either working in the UK or you’re not working because you’re:

* retired
* pregnant or looking after children
* sick or disabled or your partner has died

- Condition to Check: Youŕe working in the UK
- Known Information: Q: Are you working in the UK? A: No
- Question: I was born in North Carolina and my husband and I are still trying to conceive. May I be able to get Child Tax Credit?
- Response:

Reasoning: The condition to check is whether the user is working in the UK. According to the known information about the user, she mentioned that she is not working in the UK. So the condition is contradicted by the known information.

Satisfaction: contradicted

Table 19: ShARC prompt example for condition verification.

- Please answer the question based on the provided document.
- Provided Document:

6. Immigration control

You may be able to get Child Tax Credit if you’re either working in the UK or you’re not working because you’re:

* retired
* pregnant or looking after children
* sick or disabled or your partner has died

- Known Information: Q: Are you working in the UK? A: No; Q: You are retired? A: No; Q: You are pregnant? A: No; Q: You are looking after children? A: No; Q: You are sick or disabled? A: Yes
- Question: I was born in North Carolina and my husband and I are still trying to conceive. May I be able to get Child Tax Credit?
- Response:

Reasoning: The user is asking whether she may be able to get Child Tax Credit. According to the documents, you may be able to get Child Tax Credit if you’re either working in the UK, or if you are not working because you’re retired, pregnant or looking after children, sick or disabled, or your partner has died. In the known information, the user mentioned that she is not working in the UK, she is not retired or pregnant or looking after children, but she is sick or disabled. So she is still eligible to get Child Tax Credit.

Answer: Yes

Table 20: ShARC prompt example for answer generation.