Learning on Structured Documents for Conditional Question Answering

Zihan Wang, Hongjin Qian, Zhicheng Dou*
Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China
Engineering Research Center of Next-Generation Intelligent Search and Recommendation, MOE
{wangzihan0527, ian, dou}@ruc.edu.cn

Abstract

Conditional question answering (CQA) is an important task in natural language processing that involves answering questions that depend on specific conditions. CQA is crucial for domains that require the provision of personalized advice or making context-dependent analyses, such as legal consulting and medical diagnosis. However, existing CQA models struggle with generating multiple conditional answers due to two main challenges: (1) the lack of supervised training data with diverse conditions and corresponding answers, and (2) the difficulty to output in a complex format that involves multiple conditions and answers. To address the challenge of limited supervision, we propose LSD (Learning on Structured Documents), a self-supervised learning method on structured documents for CQA. LSD involves a conditional question generation method and a contrastive learning objective. The model is trained with LSD on massive unlabeled structured documents and is fine-tuned on labeled CQA dataset afterwards. To overcome the limitation of outputting answers with complex formats in CQA, we propose a pipeline that enables the generation of multiple answers and conditions. Experimental results on the ConditionalQA dataset demonstrate that LSD outperforms previous CQA models in terms of accuracy both in providing answers and conditions.

1 Introduction

Recently, question answering (QA) has gained increasing interest in the field of Natural Language Processing. Various types of question answering tasks, such as knowledge-based QA (Cui et al., 2017), open domain QA (Kwiatkowski et al., 2019), and multi-hop QA (Yang et al., 2018), have been extensively studied. Among them, conditional question answering (CQA) (Sun et al., 2022a) is becoming increasingly important in various contexts, such as medical diagnosis, legal consultation, financial analysis, and more. Unlike the traditional question answering tasks that only accepts a question and returns an answer, CQA involves understanding a complex and lengthy document, finding all possible answers under different **conditions**, and determining under what **condition** the answer is applicable. Figure 1 shows an example for CQA, where the answer could be different when the questioner is under different conditions. The CQA task includes providing potential answers "yes" and "no" and their corresponding conditions based on the given question and scenario.

Previous studies on CQA can be broadly categorized into two groups: extractive and generative methods. Extractive methods (Ainslie et al., 2020) (Sun et al., 2021) extract the most relevant span from a document as answers and conditions. In contrast, generative methods (Izacard and Grave, 2021) (Sun et al., 2022b) use a generative model to generate answers along with their corresponding conditions directly. However, current CQA models face two common challenges. Firstly, the supervised data for CQA is limited and expensive to obtain. Unlike traditional QA datasets, CQA requires specific annotations that include scenarios, answers, and conditions, making the data collection process more extensive and time-consuming. Secondly, current CQA models are unable to provide multiple conditional answers in a coherent and controlled format. Extractive methods for CQA are mostly only able to provide a single answer or condition for a question, limiting their ability to produce multiple conditional answers.

*Corresponding author.

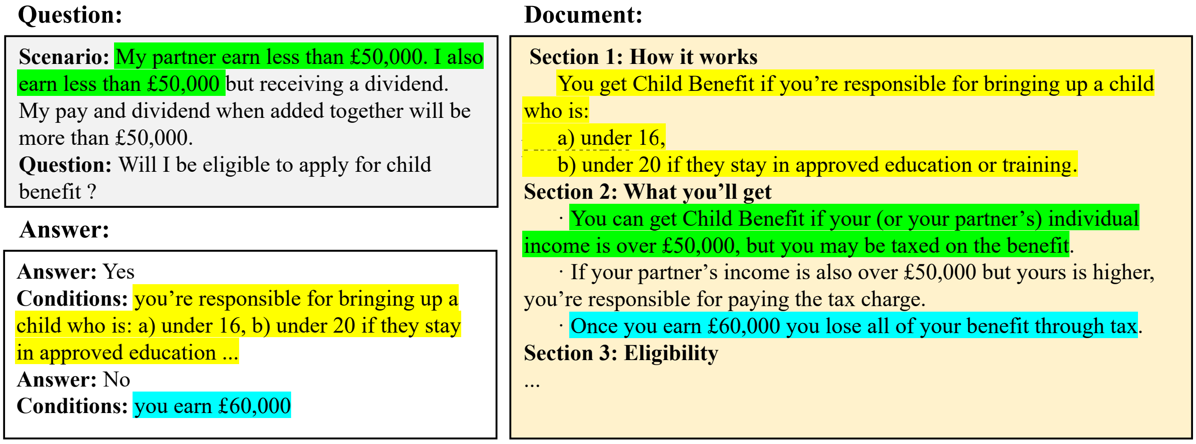

Figure 1: An example for CQA. The right side is a snapshot of a document discussing the policy of claiming Child Benefits. The green text span is the condition that has been satisfied. The yellow and blue text spans are the conditions for "Yes" and "No" respectively.

Conversely, generative methods may generate inconsistent and incoherent answers and conditions due to their inherent randomness, especially when generating multiple conditional answers. These challenges underscore the need for improved approaches to effectively handle the generation of multiple conditional answers in CQA.

In order to solve the problem of limited supervision, we propose a self-supervised learning method called LSD (Learning on Structured Documents). LSD consists of two main components: conditional question generation and contrastive learning. For conditional question generation, our intuition is that if a more precise context that contains sufficient information to answer a conditional question can be passed to the QA model, then the conditional answers given through this context will have high accuracy and can be used for subsequent training. To achieve this goal, we propose a selective extraction process that extracts parts of a structured document that are likely to be able to answer a conditional question. For a certain selected part of the document, we use a state generator to generate a conditional question and user scenario, and use a label generator to generate highly believed answers. For contrastive learning, we use four methods of document perturbation to perturb the structure of the document, including node reordering, repetition, masking, and deletion. These methods will change the content of the document but have little impact on its semantics. We design a contrastive learning objective that encourages the model to give similar representations of document corresponding sentences before and after perturbation, enabling the model to learn effective semantic representations from complex documents and helping with conditional question answering.

To solve the problem of complex output formats, we propose a pipeline that can generate multiple answers and their corresponding conditions. Our pipeline extracts answer spans from sentences, generating query vectors for each answer and key vectors for each candidate condition. Afterward, we calculate the query-key matching score for each answer and condition, and choose the best matches as the final output. Unlike existing methods, our pipeline utilizes the structure of documents to generate questions and conditions, and can generate controllable multiple conditional answers.

To verify the effectiveness of our method, we conduct experiments on the conditionalQA dataset (Sun et al., 2022a). The experimental results showed that our method outperformed all baseline models in terms of answer and condition accuracy, indicating that our method can provide accurate answers and corresponding conditions to effectively answer conditional questions.

In summary, our contributions are three-fold:

(1) We propose LSD, a self-supervised learning method for structured documents based on question generation and contrastive learning, which effectively solves the problem of insufficient supervision for conditional question answering;

(2) We propose a pipeline that generates a query and key vectors for candidate answers and conditions and matching similarity for them, which can provide controllable conditional answers;

(3) The experimental results indicate that our method can answer conditional questions more effectively compared to previous conditional question answering methods.

2 Related Work

2.1 Conditional Question Answering

Conditional question answering (CQA) has been studied using extractive and generative methods. Extractive methods, such as ETC (Ainslie et al., 2020) and DocHopper (Sun et al., 2021), use two separate models to extract answers and conditions. ETC pipeline uses two separate encoders to extract answers from supporting documents and identify conditions. DocHopper, on the other hand, iteratively attends to different sentences to predict evidences, answers and conditions step-by-step. Generative methods such as FiD (Izacard and Grave, 2021) use a single generative model to generate answers with conditions. FiD splits documents into sentences, encodes the sentences separately, and jointly decodes all encoded representations to generate answers with conditions. TReasoner (Sun et al., 2022b) is a discriminative-generative model that first checks whether each sentence could be a condition and then generates the answer with the context. However, these models suffer from several limitations, including a lack of sufficient supervised data, which can lead to overfitting and poor reasoning capabilities. Furthermore, pipeline designs have a limited ability to generate multiple hybrid-type answers and conditions. Therefore, improving the performance of CQA through a suitable pipeline is crucial, and our work aims to address these challenges.

2.2 Self-Supervised Learning

Self-supervised learning methods have gained significant traction in recent years, as they allow models to learn powerful representations without relying on large amounts of labeled data. Various language models, such as GPT-3 (Brown et al., 2020), BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), BART (Lewis et al., 2020), have leveraged unsupervised pre-training to achieve remarkable results on extensive natural language tasks. There have also been multilingual approaches like XLM (Conneau et al., 2020), unsupervised machine translation (Lample et al., 2018), question generation techniques such as QA-based multiple-choice question generation (Le Berre et al., 2022), Web-pretraining (Guo et al., 2022), and deep reinforcement learning (Chen et al., 2019). On the other hand, contrastive learning has emerged as a powerful method for representation learning, with models like SimCSE (Gao et al., 2021), ELECTRA (Clark et al., 2020), DPR-QA (Karpukhin et al., 2020) and XMOCO (Yang et al., 2021) achieving state-of-the-art results across various natural language understanding and generation tasks by learning to distinguish between semantically similar and dissimilar inputs.

3 Preliminaries: Structured Documents

Structured documents contain complex and rich structural information, which is beneficial for learning conditional question answering. In this work, our model is trained on HTML documents, a widely used type of structured document. HTML documents are easily accessible and often contain rich semantic information, including tables, lists, and more. The underlying structure of an HTML document is represented by the Document Object Model (DOM) tree, wherein the entire document constitutes the root node, and individual elements are organized as child nodes within the hierarchy.

A diagram of a DOM tree is shown in Figure 2. Since HTML does not always demonstrate a clear hierarchy among elements, we adopt a tag precedence order to convert HTML documents into trees, thus making the relationships between elements explicit. We order commonly used tags as: ⟨title⟩ - ⟨h⟩ - ⟨p⟩ - ⟨li⟩ / ⟨tr⟩. Each node's parent is the closest preceding higher-level node. For example, the ⟨h1⟩ tag is a section title and is the parent of ⟨h2⟩ subsection titles. The ⟨h2⟩ tag is a subheading and is the parent of ⟨p⟩ text elements. We omit tags that do not contain important information, such as ⟨b⟩ (bold),⟨i⟩ (italic), and ⟨a⟩ (hyperlink) tags. With our approach, each sentence within the HTML document can be clearly represented as a node in the document tree.

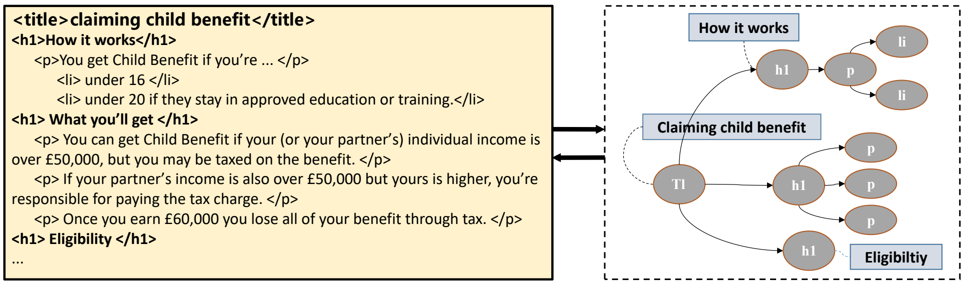

Figure 2: An example of the schematic diagram of a DOM tree. HTML tags can be used to create a hierarchy of sentences in a document, with some tags considered more senior than others. The nearest former superior tag of a node is its parent node.

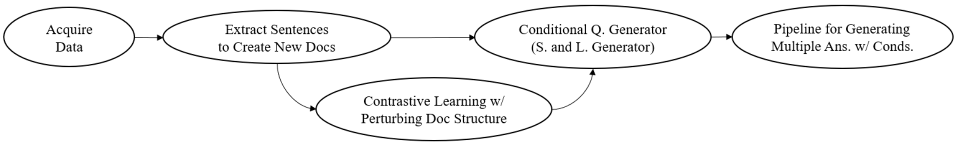

Figure 3: An overall illustration of our approach.

To compile a corpus of structured documents for the CQA task, we consider the following criteria:

- Logical Structure: Documents should possess clear logical structures, including specific conditions and provisions, to facilitate conditional reasoning in the CQA task
- Standardized Format: Documents should adhere to a standardized HTML format with minimal noise, such as advertisements.
- Data Quality: The corpus should comprise formal, authoritative, and reviewed documents to ensure data reliability and accuracy.

Based on these criteria, we propose to train our model to learn on **national government websites**, which are known for their formal and authoritative nature. We conduct web scraping to gather documents, filtering for policy documents, laws and regulations, and administrative guidelines, as they tend to exhibit clear logical structures and contain specific conditions relevant to the CQA task. For additional details regarding the construction of the corpus, which is referred to as DATASET, please refer to Appendix A.

4 Our Approach

In this section, we will introduce our proposed method LSD, which includes a conditional question generation module and a contrastive learning method for self-supervised learning on structured documents. After that, we will illustrate our pipeline that generates multiple conditional answers by calculating the matching score of answers and candidate conditions with query and key vectors. The overall process of our method is shown in Figure 3.

4.1 Decomposed Conditional Question Generation with Document Extraction

Let the conditional question generator be G and the conditional question answering model be M. Recall that the intuition of our approach is that if we can provide G with a more precise context with sufficient information for a conditional question, then G can answer the question correctly, and the obtained question-answer data can be used to train M. To achieve this, we adopt a two-step method: selective extraction and question generation. A specific overview of conditional question generation is in Algorithm 1.

4.1.1 Selective Extraction

Selective extraction aims for precise context to generate conditional questions. The main requirement for the selected context is to contain sufficient information to answer a conditional question. To guide our

Algorithm 1 Conditional Question Generation

Require: Structured doc set DATASET
Ensure: Cond. question q, scenario sc, answer a, condition c
procedure QUESTIONGEN(DATASET)
  Init: state gen. G_S, label gen. G_L
  Sample doc D from DATASET
  Select non-leaf text node $s \in D$ as potential answer
  Construct extracted $\overline{D}$ by selecting anc., child., sibl., and sibl. child. of s
  Gen. question q, scenario sc using $G_S(\overline{D})$
  Gen. cond. answers $A = (a_i, c_i)$ using $G_L(q, sc, \overline{D})$
end procedure

	answers	conditions
leaf node	86.93%	92.53%
text node	92.49%	98.33%

(a) Features of answers and condition nodes: whether they are leaf nodes or text nodes.

	a-a pairs	c-c pairs	a-c pairs
sibling-sibling	66.55%	53.67%	-
parent-child	-	-	39.59%

(b) Features of answer and condition pairs: answer pairs (a-a), condition pairs (c-c), and answer-condition pairs (a-c).

Table 1: Statistics of the ConditionalQA train dataset for guiding selective extraction.

extraction strategy, we analyzed the ConditionalQA dataset, which also leverages structural documents for the CQA task. (Table 1). We analyzed the occurrence and correlations between answers and conditions, and observed several features: (1) answers and conditions are mainly located in leaf text nodes, such as ⟨p⟩ and ⟨li⟩ nodes; (2) answers are usually siblings; (3) conditions for extractive answers may be their child nodes; (4) sibling nodes with the same parent node can serve as parallel answers.

Guided by these insights, our extraction method involves the following steps. Firstly, we randomly select a non-leaf text node as a potential answer, because conditional answers are most likely to be such nodes. Then, we then extract its ancestors, children, siblings, and their children from the document tree, because: (1) ancestor nodes provide the macro context of higher-level topics; (2) child nodes offer potential conditions; (3) siblings, along with their children, provide parallel answers. Afterward, we obtain an extracted document that enables generating conditional questions aligned with the original text and answerable with accuracy.

4.1.2 Question Generation

The question generation approach are decomposed into two tasks: state generation and label generation. The first task is to generate question q and scenario sc given the extracted structured document $\overline{D}$, and the second task is to generate highly accurate conditional answers $A = \{(a_1, c_1), (a_2, c_2), \ldots\}$, where a_i is an answer and c_i denotes the corresponding conditions. We leverage a state generator G_S, a sequence-to-sequence (Sutskever et al., 2014) generative model to provide diverse content, and a conditional answer extraction model G_L, an extractive model to provide accurate answers. More information on the network structure and training process of G can be found in Appendix C.

In general, by leveraging structured documents for precise document extraction and supervised generator training, we ensure that we can identify the locations of potential answers and conditions within structured documents, thereby achieving the generation of high-quality conditional questions and ensuring the correct answering of questions for subsequent training.

4.2 Purturbation-based Document Contrastive Learning

Our contrastive learning approach on structured documents involves the following steps: document perturbation, positive sample generation, and contrastive loss computation. At the training stage, the computed loss is added to the total training loss for optimization.

Operation	Description	Advantages
Node masking	Mask node with [MASK] of same length	Focus on structure & context
Node deletion	Delete non-root node & descendants	Learn node dependencies & importance
Node cloning	Clone node & descendants as another child	Identify semantically similar elements
Node shuffling	Shuffle child nodes within parent	Understand impact of node order

Table 2: Basic operations for Contrastive Learning.

4.3 Document Perturbation

To perturb the original document D and obtain a perturbed document $\hat{D}$, we introduce a set of basic operations T that can be applied to the document structure. These operations, detailed in Table 2, include node masking, node deletion, node cloning, and node shuffling. Assume the original document D has a title s_0 and m nodes $(n_1, n_2, ..., n_m)$. Starting with the original document D_0, we apply k random operations from the set T to generate the perturbed document $\hat{D} = D_k$. Each operation T_i is applied as $T_i(D_j) = D_{j+1}$ for any T_i selected from T.

4.3.1 Positive Pair Generation

We get positive pairs from D and $\hat{D}$ for loss calculation. For the i^{th} node n'_i in the perturbed document $\hat{D}$, there is a corresponding source node n_{k_i} in the original document D. We form positive pairs using tags t'_i and t_{k_i} that serves as global tokens of the nodes, which effectively convey node type and semantics despite structural changes during document perturbation.

4.3.2 Contrastive Loss Computation

We use the InfoNCE loss $\mathcal{L}_{CL}(D, \hat{D})$ for contrastive learning, defined as:

$$\mathcal{L}_{CL}(D, \hat{D}) = \sum_{i=1}^{m'} \frac{e^{\text{sim}(t'_i, t_{k_i})}}{e^{\text{sim}(t'_i, t_{k_i})} + \sum_{t^-_{k_i}} e^{\text{sim}(t'_i, t^-_{k_i})}}, \tag{1}$$

where m' is the total number of nodes in $\hat{D}$, t'_i and t_{k_i} represents a positive pair, and $t^-_{k_i}$ represents tags of any nodes other than n_{k_i} in D. sim computes the similarity between tags using the dot product of their hidden states from a neural document encoder, detailed in 4.3. The loss encourages high similarity between each t'_i and t_{k_i} while minimizing similarity with negative tags $t^-_{k_i}$.

In general, our contrastive learning approach enables self-supervised training by perturbing structured documents to construct contrastive pairs. By reinforcing node correspondence in structured documents, the method supports conditional question answering models in accurately capturing semantic connections between conditions and answers in complex contexts.

4.4 Pipeline for Answering Conditional Questions

Our proposed pipeline, illustrated in Figure 4, comprises three steps: (1) document encoding, (2) multiple answer extraction, (3) condition determination. An auxiliary task Evidence Node Finding is added when necessary (Appendix D).

4.4.1 Document Encoding

In the document encoding process, we first construct the input sequence, which consists of special tokens "[yes]" and "[no]" document content, question, and scenario. The special tokens are used to represent affirmative/negative answers. We represent the input sequence as follows:

$$\begin{aligned}\text{Input} = \text{“[yes][no]document : ”} + D \\ + \text{“question : ”} + q + \text{“Scenario : ”} + sc,\end{aligned}$$

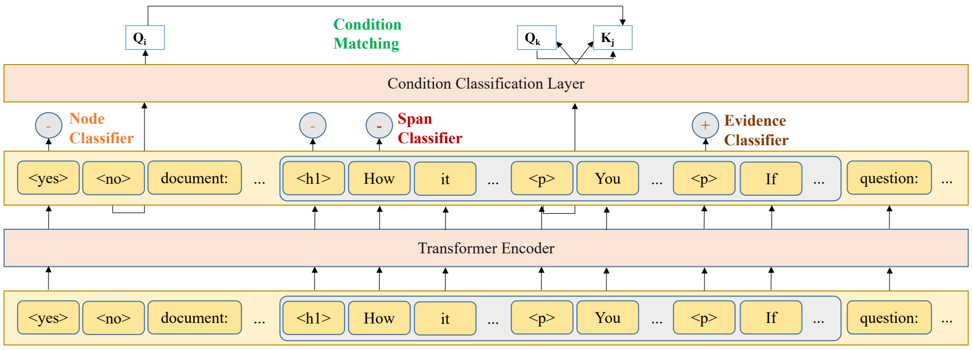

Figure 4: Our pipeline to answer conditional questions.

where [yes] and [no] are special tokens for yes / no answers. It is passed to E returning hidden states:

$$\begin{aligned}\text{Output} &= \text{Transformer}(\text{Input}) \\ &= h_{[\text{yes}]}, h_{[\text{no}]}, ...h_{t_i}, h_{a_{ij}}, ...,\end{aligned}$$

where $h_{[\text{yes}]}, h_{[\text{no}]}$ are hidden states of special tokens, h_{t_i} represents hidden state of the tag of the i^{th} node in the document, and $h_{a_{ij}}$ represents hidden state of the i^{th} node's j^{th} token. These hidden states are used by the multi-layer perceptron (MLP) classifiers P_S, P_N, P_V to calculate probabilities for answer extraction and condition determination.

4.4.2 Multiple Answer Extraction

To simplify the answer extraction process, we assume that a node has no more than one answer, and we retain only one answer if multiple exist. Since it's rare that a node has multiple answers, this process simplifies extraction by identifying potential answer nodes and determining the answer's start and end positions within the node.

We use two classifiers: a node classifier P_N to identify answer-containing nodes (or yes/no tokens) and an answer span classifier P_S to locate the answer's position within selected nodes.

For node classification, we set:

$$\begin{aligned}p^N_{\text{yes/no}} &= P_N(h_{[\text{yes}]/[\text{no}]}), \\ p^N_i &= P_N(h_{t_i}),\end{aligned} \tag{2}$$

where p represents probabilities given by these classifiers. From the above, we can obtain yes/no answers and sentences containing extractive answers from node classification results. At training, We set a Binary Cross Entropy (BCE) loss for node classification:

$$\mathcal{L}_{\text{bool}} = \frac{\text{BCE}(p^N_{\text{yes}}, \mathbb{I}^N_{\text{yes}}) + \text{BCE}(p^N_{\text{no}}, \mathbb{I}^N_{\text{no}})}{2}, \tag{3}$$

$$\mathcal{L}_{\text{extractive}} = \frac{1}{m}\sum_{i=1}^{m} \text{BCE}(p^N_i, \mathbb{I}^N_i), \tag{4}$$

$$\mathcal{L}_{\text{N}} = \mathcal{L}_{\text{bool}} + \mathcal{L}_{\text{extractive}}, \tag{5}$$

where $\mathbb{I}$ represents boolean labels to indicate whether the given element satisfies some requirements, e.g., $\mathbb{I}^N_i$ represents whether the i^{th} node is a potential answer node, assuming totally m nodes.

For answer span localization, we adopt a span locator P_S for any positive nodes of the above process by:

$$\begin{aligned}p^{S_i}_{j1}, p^{S_i}_{j2}, ... &= P_{S_i}(a^A_{j1}), P_{S_i}(a^A_{j2}), ..., \\ &(i \in (1, 2), j \in (1, 2, ..., k)),\end{aligned} \tag{6}$$

where P_{S_1}, P_{S_2} predict start / end of the answer, a^A_{ju} denotes the u^{th} token of the j^{th} predicted node n^A_j to have an answer, and $p^{S_i}_{ju}$ are the predicted probabilities. At training, we adopt a span loss:

$$\mathcal{L}_{\text{S}} = \frac{1}{2k_r} \sum_{i=1}^{2} \sum_{j=1}^{k_r} \sum_{u=1}^{l_{n^A_j}} \frac{1}{l_{n^A_j}} \text{BCE}(p^{S_i}_{ju}, \mathbb{I}^{S_i}_{ju}), \tag{7}$$

where k_r represents the real count of answers and $l_{n^A_j}$ represents the number of tokens of n^A_j.

4.4.3 Condition Determination

To align with the document structure, we define that a potential condition must be a node in the document. Therefore, the condition determination process is to predict the probability of a node being the condition of an answer. To model this, we assign query vectors to answers, and key vectors to nodes:

$$\begin{aligned} h^Q_i &= W^Q \,\text{ReLU}(W^H h_i), \\ h^K_j &= W^K \,\text{ReLU}(W^H h_j), \end{aligned} \tag{8}$$

where h_i, h_j denotes the hidden state of i^{th} answer and j^{th} sentence. W^H, W^Q, W^K are transformation matrices, h^Q_i, h^K_j denotes the query vector of i^{th} answer and the key vector of j^{th} sentence.

Then, we calculate on conditions:

$$p^C_{ij} = \text{sigmoid}(h^Q_i \cdot h^K_j), \tag{9}$$

where p^C_{ij} denotes the probability of j^{th} node to be the condition of the i^{th} answer. We adopt the following loss for training:

$$\mathcal{L}_{\text{C}} = \frac{1}{k_r m} \sum_{i=1}^{k_r} \sum_{j=1}^{m} \text{BCE}(p^C_{ij}, \mathbb{I}^C_{ij}). \tag{10}$$

From the above process, we can fuse the representations of answers and conditions to model the condition determination process. Therefore, our pipeline has resolved the conditional question answering task. At training, we linearly mix up all losses mentioned:

$$\mathcal{L}_{\text{train}} = \mathcal{L}_{\text{N}} + \mathcal{L}_{\text{S}} + \mathcal{L}_{\text{C}} + \mathcal{L}_{\text{CL}}. \tag{11}$$

5 Experiments

5.1 Datasets and Evaluation Metrics

To construct a dataset of structured documents, we scrape web pages from English websites. Our data collection process is detailed in Appendix A. To evaluate LSD's effectiveness on CQA, we conduct experiments on ConditionalQA (Sun et al., 2022a) dataset. It consists of extractive questions, yes / no questions, and not-answerable questions. The task is to find all answers with corresponding conditions on a structured document based on the given questions and scenarios.

Evaluation To evaluate model performance, we adopt the metrics of EM / F1 and EM / F1 with conditions, which are introduced in the ConditionalQA (Sun et al., 2022a) dataset. EM / F1 are conventional metrics, and EM / F1 with conditions jointly measures the correctness of the answer and the predicted conditions. For not answerable questions, EM and F1 are 1.0 if and only if no answer is predicted.

5.2 Results

We compared the LSD model with all of the baseline models for CQA. To evaluate the model's performance in both answering questions and providing conditions, we present results for the entire ConditionalQA dataset and its subset of conditional questions.

	Yes / No		Extractive		Conditional		Overall	
	EM / F1	w/ conds	EM / F1	w/ conds	EM / F1	w/ conds	EM / F1	w/ conds
ETC-pipeline	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
TReasoner	**73.2 / 73.2**	**54.7 / 54.7**	34.4 / 48.6	30.3 / 43.1	51.6 / 56.0	12.5 / 14.4	57.2 / 63.5	**46.1 / 51.9**
LSD (ours)	71.6 / 71.6	51.6 / 51.6	**39.9 / 56.4**	**31.6 / 43.8**	**57.3 / 61.8**	**21.4 / 25.1**	**58.7 / 66.2**	45.0 / 50.5

Table 3: The results of our experiments on the ConditionalQA dataset. "EM / F1" shows the standard EM / F1 metrics based on the answer span only. "w/ conds" shows the conditional EM / F1 metrics introduced in (Sun et al., 2022a). The results for the baseline models are taken from (Sun et al., 2022a) (Sun et al., 2022b)

	Answer (w / conds)	Conditions (P / R / F1)
ETC-pipeline	/	/
DocHopper	/	/
FiD	3.2 / 4.6	98.3 / 2.6 / 2.7
FiD (cond)	6.8 / 7.4	12.8 / 63.0 / 21.3
TReasoner	10.6 / 12.2	34.4 / 40.4 / 37.8
LSD (ours)	**21.4 / 25.1**	**69.3 / 39.4 / 50.2**

Table 4: Experimental results on the subset of questions in ConditionalQA (dev) with conditional answers. Results of the baseline models are obtained from (Sun et al., 2022a) (Sun et al., 2022b). The first two models "do not provide any conditions when they achieved the best performance on the overall dataset".

5.2.1 Main Result

Table 3 shows the results on the entire conditionalQA dataset. The result indicates that:

(1) LSD outperforms all baselines in EM / F1 and conditional EM / F1 for extractive and conditional questions, demonstrating the effectiveness of our conditional question generation and contrastive learning.

(2) LSD performs not as well as TReasoner in Yes / No questions. We speculate that its attributed to LSD inclination to provide conditional answers due to training with our question generation system (Appendix B), which is penalized by the evaluation metric in (Sun et al., 2022a).

(3) In "w/ conds" overall results, LSD performs less well than TReasoner, potentially due to TReasoner's specialized multi-hop reasoning for condition determination, which may warrant further enhancement in LSD.

5.2.2 Conditional Accuracy

To further evaluate our model's ability to provide conditions for answers, we additionally report results on the subset of conditional questions in Table 4. We evaluate the results using the "w/ conds" metric, as well as precision, recall, and F1 of retrieved conditions for conditional answers. The result shows that our method significantly outperforms the current model in providing conditions.

6 Analysis

In this section, we conduct an ablation study to investigate the impact of our document modeling designs and contrastive learning. We further analyze the question generation process by evaluating the quality of generated questions and the accuracy of generated labels.

6.1 Ablation Study

We conduct an ablation study on the dataset to investigate the impact of conditional question generation and contrastive learning. Results on the dev set of ConditionalQA in Table 5 show that both conditional

	Yes / No		Extractive		Conditional		Overall	
	EM / F1	w/ conds	EM / F1	w/ conds	EM / F1	w/ conds	EM / F1	w/ conds
LSD (ours)	**71.6 / 71.6**	**51.6 / 51.6**	**39.9 / 56.4**	**31.6 / 43.8**	**57.3 / 61.8**	**21.4 / 25.1**	**58.7 / 66.2**	**45.0 / 50.5**
w/o CL	69.6 / 69.6	49.9 / 49.9	38.0 / 55.7	29.8 / 43.2	54.6 / 59.1	19.4 / 23.2	56.9 / 64.8	43.3 / 49.4
w/o QG	67.9 / 67.9	47.1 / 47.1	37.2 / 54.9	29.0 / 42.5	54.0 / 58.6	17.8 / 21.6	55.7 / 63.7	41.6 / 47.6

Table 5: Ablation study of our model on the dev set of ConditionalQA.

	ROUGE (%)	BLEU (%)
question	42.07	38.19
scenario	39.57	41.65

(a) Evaluation on state generator's output quality.

	Yes / No	Extractive	Conditional	Overall
EM / F1 (%)	79.6 / 79.6	51.2 / 67.2	69.9 / 73.8	67.8 / 75.0
w / conds (%)	50.8 / 50.8	38.9 / 51.3	33.4 / 35.5	47.9 / 53.4

(b) Evaluation on accuracy of generated labels.

Table 6: Evaluation on our question generation method.

question generation and contrastive learning are of importance, as removing either of them causes a significant performance drop in the final results.

6.2 State Generator's Output Quality

We use BLEU and ROUGE-L to measure the state generator's generated questions and scenarios' similarity to questions and scenarios from the evaluation dataset for question generation, QG-dev (detailed in Appendix C). The results are shown in Table 6a. Some examples are shown in Appendix E.

6.3 Label Generator's Output Accuracy

We evaluate our label generator's capability in providing accurate answers for questions given the extracted documents from QG-dev, shown in Table 6b. The result shows that the label generator can provide accurate answers given a selected context from the document.

7 Conclusion and Limitations

In this paper, we present Learning on Structured Documents (LSD), a self-supervised learning method for conditional question answering. LSD uses a conditional question generation method to leverage massive structured documents while improving conciseness, and applies contrastive learning to learn effective semantic representations from complex documents. We further propose a pipeline that could generate multiple answers and conditions to better handle the CQA task. We verify the effectiveness of the proposed method on the ConditionalQA dataset. For future work, we plan to investigate how to better generate conditional questions and improve our model's performance in providing correct answers.

Despite the effectiveness of LSD in utilizing the structure of massive unsupervised data, there are still some potential points for improvement. One issue is that the state generator is only trained on answerable questions, leading to a distribution bias that there might be unanswerable questions. In addition, our pipeline can still not handle the position where a sentence has more than one answer, which limits our model's performance for broader scenarios. We will resolve these issues in future work.

Acknowledgements

This work was supported by National Natural Science Foundation of China No. 62272467, Beijing Outstanding Young Scientist Program No. BJJWZYJH012019100020098, and Public Computing Cloud, Renmin University of China. The work was partially done at Beijing Key Laboratory of Big Data Management and Analysis Methods.

References

Joshua Ainslie, Santiago Ontañón, 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.

In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 268–284. Association for Computational Linguistics.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *CoRR*, abs/2004.05150.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Yu Chen, Lingfei Wu, and Mohammed J. Zaki. 2019. Natural question generation with reinforcement learning based graph-to-sequence model. *CoRR*, abs/1910.08832.

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

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8440–8451. Association for Computational Linguistics.

Wanyun Cui, Yanghua Xiao, Haixun Wang, Yangqiu Song, Seung-won Hwang, and Wei Wang. 2017. KBQA: learning question answering over QA corpora and knowledge bases. *Proc. VLDB Endow.*, 10(5):565–576.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 6894–6910. Association for Computational Linguistics.

Yu Guo, Zhengyi Ma, Jiaxin Mao, Hongjin Qian, Xinyu Zhang, Hao Jiang, Zhao Cao, and Zhicheng Dou. 2022. Webformer: Pre-training with web pages for information retrieval. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 15021512, New York, NY, USA. Association for Computing Machinery.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In Paola Merlo, Jörg Tiedemann, and Reut Tsarfaty, editors, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 874–880. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6769–6781. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Trans. Assoc. Comput. Linguistics*, 7:452–466.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Guillaume Le Berre, Christophe Cerisara, Philippe Langlais, and Guy Lapalme. 2022. Unsupervised multiple-choice question generation for out-of-domain Q&A fine-tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 732–738, Dublin, Ireland, May. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for 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. *CoRR*, abs/1907.11692.

Haitian Sun, William W. Cohen, and Ruslan Salakhutdinov. 2021. End-to-end multihop retrieval for compositional question answering over long documents. *CoRR*, abs/2106.00200.

Haitian Sun, William W. Cohen, and Ruslan Salakhutdinov. 2022a. Conditionalqa: A complex reading comprehension dataset with conditional answers. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 3627–3637. Association for Computational Linguistics.

Haitian Sun, William W. Cohen, and Ruslan Salakhutdinov. 2022b. Reasoning over logically interacted conditions for question answering. *CoRR*, abs/2205.12898.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

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. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics.

Nan Yang, Furu Wei, Binxing Jiao, Daxing Jiang, and Linjun Yang. 2021. xMoCo: Cross momentum contrastive learning for open-domain question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6120–6129, Online, August. Association for Computational Linguistics.

Yang You, Jing Li, Sashank J. Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. 2020. Large batch optimization for deep learning: Training BERT in 76 minutes. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Appendix

A DATASET curation details

	UK	US	CA	Overall
count	17,881	577	12,115	30,573
Avg. w	709	179	2,538	1423
Avg. s	54	26	128	83
Avg. w/s	12.9	6.9	19.8	17.0
Tag dist.	14:45:41	38:40:22	10:40:50	12:41:57

Table 7: Statistics of our scraped dataset. We present document count, average document length measured by word (Avg. w) and sentences (Avg. s), average sentence length (Avg w/s) and tag distribution (h:p:li/tr).

DATASET contains a total of 30,573 documents, approximately 362MB in size (1×10^8 tokens). The statistics of our scraped dataset are shown in Table 7. The data curation process are detailed below.

A.1 Data Acquisition

To build DATASET, we scrape web pages from government websites: https://www.gov.uk, https://www.ca.gov, and https://www.usa.gov, as they have professional English material and have a massive number of well-structured documents, such as policies, regulations, and proposals.

A.2 Data Filtering

Page Category Filtering We use automated web scraping to categorize pages on the selected government websites based on URL. We retaine only pages related to policy documents, regulatory provisions, administrative guidelines, etc.
Content Validity Check We further examined the retained pages to exclude invalid, redundant, or duplicate documents.

A.3 Data Cleaning

Tag Normalization We use automated cleaning and standardization tools to fix irregular HTML tags and attributes in documents, close unclosed tags, and standardize attribute values.
Irrelevant Content Removal We remove nodes without text, advertisements, hyperlinks, images, videos, and other irrelevant information, retaining textual content for better model understanding of document structure and content.
Node Filtering We filtere nodes containing document content, i.e., <h1> to <h6> (headings), <p> (paragraphs), <li> (list items), <tr> (table rows), etc.
DOM Tree Construction We use an HTML parser to parse the filtered nodes and construct the Document Object Model (DOM) tree following the method proposed in section 3.

A.4 Dataset Splitting

We split the processed dataset into training and validation sets for model training and performance evaluation with a ratio of 4:1.

B Question Generation details

We present the statistics to show our question generation module's behavior on the scraped augmentation corpus. We randomly generate 1,000 samples with the QG module and present results in Table 8.

Our Dataset	Yes / No	Extractive	Conditional
Percentage	52.4	47.5	45.1
Avg. answer	1.36	1.46	1.86
Avg. condition	0.89	1.04	2.14
Avg. context	292	350	413
Avg. document	1,467	1,260	1,525
ConditionalQA	Yes / No	Extractive	Conditional
Percentage	51.1	44.6	23.4
Avg. document		1358	

Table 8: Statistics of our generated dataset and ConditionalQA dataset in comparison. We present the percentage of every type of questions, average answer count, condition count, condition count, context length and document length (by word) if applicable.

C Implementation Details

C.1 Network Structure and Setup

For the conditional question generator G: we adopt BART [1] (Lewis et al., 2020), a seq-to-seq transformer for state generator G_S; for label generator G_L, we adopt the same setting of M, as detailed below.

For conditional question answering model M: We adopt Longformer [2] (Beltagy et al., 2020), a Transformer designed for long complex context, for the neural document encoder E; for MLP classifiers P_N, P_S, P_V, we set num_layers=2 and dim_hidden_states=768; for transformation matrices, we set dim$(W_H) = (3072, 768)$ and dim(W^Q) = Dim$(W^K) = (768, 3072)$.

To setup Longformer, we set the HTML tags as its global tokens. For extremely long documents beyond length limit, we chunk them into pieces with overlap and aggregate predicted answers from these pieces.

C.2 Training Conditional Question Generator

To train conditional question generator G, we use 80% data of the ConditionalQA train set, named QG-train, and the rest for evaluation, named QG-dev. We take the descendants and ancestors of all given evidence sentences from the document for extraction. We train G on QG-train for 10 epochs, adopting the Adam (Kingma and Ba, 2015) optimizer, setting learning rate to 3e-5 and batch size to 32.

C.3 Training Conditional Question Answerer

Training conditional question answering model M consists of two stages. In the self-supervised stage, we train M on our scraped dataset for 20 epochs, with a newly generated question and answer data for every epoch. We use the LAMB (You et al., 2020) optimizer for this stage, with the learning rate set to 1e-4 and the batch size set to 256. In the supervised stage, we adopt the Adam (Kingma and Ba, 2015) optimizer, setting the learning rate to 3e-5 and batch size to 32, and trained on ConditionalQA train set for 50 epochs. For both stages of training, we adopt a warm-up episode of 10% proportion with linear learning rate decay. For document chunking, We set the maximum of document length to 2000 to fit into the GPU memory, with an overlap of 100 tokens. For contrastive learning, we adopt k=5.

D Auxiliary Task: Evidence Node Finding

To improve model reasoning for yes / no questions, we introduce an auxiliary task to identify evidence nodes supporting the answer. The task is jointly trained with others and is active when datasets provide evidence information. We use an evidence classifier P_V for this task and define:

$$p_i^V = P_V(\mathrm{h_{t_i}}), \tag{12}$$

[1]https://huggingface.co/facebook/bart-large

[2]https://huggingface.co/allenai/longformer-large-4096

$$\mathcal{L}_{\text{E}} = \frac{1}{m}\sum_{i=1}^{m}\text{BCE}(p_i^V, \mathbb{I}_i^V), \tag{13}$$

When the evidence node finding task is activated, the training loss turns to:

$$\mathcal{L}_{\text{train}} = \mathcal{L}_{\text{N}} + \mathcal{L}_{\text{S}} + \mathcal{L}_{\text{C}} + \mathcal{L}_{\text{CL}} + \mathcal{L}_{\text{E}}. \tag{14}$$

E Case Studies for Question Generation

In this section, we provide 3 generated questions and the model-generated questions and conditional answers.

Example 1

Context:
⟨title⟩Funding Opportunities⟨/title⟩
⟨h1⟩Funding Opportunities⟨/h1⟩
⟨h2⟩Current Opportunities⟨/h2⟩
⟨p⟩Winter Collaborative Networks⟨/p⟩
⟨li⟩GC Key access⟨/li⟩
⟨li⟩SecureKey Concierge (Banking Credential) access⟨/li⟩
⟨li⟩Personal Access Code (PAC) problems or EI Access Code (AC) problems⟨/li⟩
⟨li⟩Social Insurance Number (SIN) validation problems⟨/li⟩
⟨li⟩Other login error not in this list⟨/li⟩

Generated Contents:
Question:
What type of access is required for the Winter Collaborative Networks funding opportunity?

Scenario:
I am interested in applying for the Winter Collaborative Networks funding opportunity, but I am unsure of what type of access is required.

Answers and Conditions:
[GC Key access, None],
[SecureKey Concierge (Banking Credential) access, None],
[Personal Access Code (PAC) problems or EI Access Code (AC) problems, None],
[Social Insurance Number (SIN) validation problems, None],
[Other login error not in this list, None]

Example 2, 3

Context:
⟨title⟩Claim Capital Allownces⟨/title⟩
⟨h1⟩What you can claim on⟨/h1⟩
⟨p⟩You can claim capital allowances on items that you keep to use in your business - these are known as plant and machinery.⟨/p⟩
⟨p⟩In most cases you can deduct the full cost of these items from your profits before tax using annual investment allowance (AIA).⟨/p⟩
⟨p⟩If youre a sole trader or partnership and have an income of č150,000 or less a year, you may be able to use a simpler system called cash basis instead.⟨/p⟩
⟨h2⟩What does not count as plant and machinery⟨/h2⟩
⟨p⟩You cannot claim plant and machinery allowances on:⟨/p⟩
⟨li⟩things you lease (unless you have a hire purchase contract or long funding lease) - you must own them⟨/li⟩
⟨li⟩items used only for business entertainment, for example a yacht or karaoke machine⟨/li⟩

⟨li⟩land⟨/li⟩
⟨li⟩structures, for example bridges, roads, docks⟨/li⟩
⟨li⟩buildings, including doors, gates, shutters, mains water and gas systems⟨/li⟩
⟨p⟩You may be able to claim structures and buildings allowance on structures and buildings.⟨/p⟩
⟨h2⟩What counts as plant and machinery⟨/h2⟩
⟨p⟩Plant and machinery includes:⟨/p⟩
⟨li⟩items that you keep to use in your business, including cars⟨/li⟩
⟨li⟩costs of demolishing plant and machinery⟨/li⟩
⟨li⟩parts of a building considered integral, known as integral features⟨/li⟩
⟨li⟩some fixtures, for example, fitted kitchens or bathroom suites⟨/li⟩
⟨li⟩alterations to a building to install plant and machinery - this does not include repairs⟨/li⟩
⟨p⟩Claim repairs as business expenses if youre a sole trader or partnership - deduct from your profits as a business cost if youre a limited company.⟨/p⟩
⟨h3⟩Integral features⟨/h3⟩
⟨p⟩Integral features are:⟨/p⟩
⟨li⟩lifts, escalators and moving walkways⟨/li⟩
⟨li⟩space and water heating systems⟨/li⟩
⟨li⟩air-conditioning and air cooling systems⟨/li⟩
⟨li⟩hot and cold water systems (but not toilet and kitchen facilities)⟨/li⟩
⟨li⟩electrical systems, including lighting systems⟨/li⟩
⟨li⟩external solar shading⟨/li⟩
⟨h3⟩Fixtures⟨/h3⟩
⟨p⟩You can claim for fixtures, for example:⟨/p⟩
⟨li⟩fitted kitchens⟨/li⟩
⟨li⟩bathroom suites⟨/li⟩
⟨li⟩fire alarm and CCTV systems⟨/li⟩
⟨p⟩You can claim if you rent or own the building, but only the person who bought the item can claim.⟨/p⟩
⟨p⟩When you buy a building from a previous business owner you can only claim for integral features and fixtures that they claimed for.⟨/p⟩
⟨p⟩You must agree on the value of the fixtures with the seller. If you do not you cannot claim for them. Agreeing the value also means the person selling the assets can account correctly for them.⟨/p⟩
⟨h2⟩If you let residential property⟨/h2⟩
⟨p⟩You can only claim for items to be used in a residential property if either:⟨/p⟩
⟨li⟩you run a furnished holiday lettings business⟨/li⟩
⟨li⟩the item is to be used in the common parts of a residential building, for example, a table in the hallway of a block of flats⟨/li⟩
⟨h2⟩Care workers⟨/h2⟩
⟨p⟩There are special rules if you run a care business.⟨/p⟩

Generated Contents:
Question:
Can I claim capital allowances on a yacht used for entertaining clients?

Scenario:
I am a business owner and I want to purchase a yacht to entertain clients. I want to know if they can claim capital allowances on it.

Answers and Conditions:
[No, None]

Question:

Can I claim capital allowances on the commercial building I purchased for my business?

Scenario:

I am a business owner and I recently purchased a commercial building to use as a location for my business.

Answers and Conditions:

[No, [⟨p⟩You cannot claim plant and machinery allowances on:⟨/p⟩, ⟨li⟩buildings, including doors, gates, shutters, mains water, and gas systems⟨/li⟩]]

[Yes, ⟨p⟩You may be able to claim structures and buildings allowance on structures and buildings.⟨/p⟩]