Reasoning over Hybrid Chain for Table-and-Text Open Domain Question Answering

Anonymous ACL submission

Abstract

Tabular and textual question answering requires systems to perform reasoning over heterogeneous information, considering table structure, and the connections among table and text. In this paper, we propose a_CHAIN-centric Reasoning and Pre-training framework (CARP). CARP utilizes hybrid chain to model the explicit intermediate reasoning process across table and text for question answering. We also propose a novel chain-centric pre-training method, to enhance the pre-trained model in identifying the cross-modality reasoning process and alleviating the data sparsity problem. This method constructs the large-scale reasoning corpus by synthesizing pseudo heterogeneous reasoning paths from Wikipedia and generating corresponding questions. We evaluate our system on OTT-QA, a large-scale table-and-text open-domain question answering benchmark, and our system achieves the state-of-the-art performance. Further analyses illustrate that the explicit hybrid chain offers substantial performance improvement and interpretability of the intermediate reasoning process, and the chain-centric pre-training boosts the performance on the chain extraction. 1

1 We will release our code and data upon acceptance.

1 Introduction

Open domain question answering (Joshi et al., 2017; Dunn et al., 2017; Lee et al., 2019) requires systems to retrieve and perform reasoning over supported knowledge, and finally derive an answer. Generally, the real-world knowledge resource is heterogeneous, which involve both semi-structured web tables and unstructured text like Wikipedia passages. Therefore, question answering over hybrid tabular and textual knowledge is essential and attracts wide attentions (Chen et al., 2020a), and is more challenging as systems need to aggregate information in both table and text considering their connections and the table structure.

As the example shown in Fig. 1, the complete reasoning process for answering the question involves hybrid information pieces in both the table (“Year” and “Points” columns in the first row) and the passage (“COVID-19”). Therefore, modeling the structural connections inside heterogeneous knowledge is critical for modeling the reasoning process. Many recent works on table-and-text open domain QA simply take the supported flattened table and passages (Chen et al., 2020a; Li et al., 2021) as a whole for question answering, which neglects the structural information and connections among table and text, and leads to more noise as full tables always contain redundant information. Secondly, these methods tackle the whole reasoning process as a black box, and lack the interpretability of the intermediate reasoning process. Moreover, the data sparsity problem is also severe, as the high-quality annotated reasoning process is hard to be obtained.

To tackle these challenges, we propose a_CHAIN-centric Reasoning and Pre-training framework (CARP), which models the intermediate reasoning

Figure 1: An example of the table-and-text QA with intermediate reasoning process. The answer is 25.3.
process across table and text with a hybrid chain for question answering. CARP first formulates a heterogeneous graph, whose nodes are information pieces in the relevant table and passages, to represent the interaction residing in hybrid knowledge. Then, it identifies the most plausible reasoning path leading to the answer with a Transformer-based extraction model. Moreover, to augment the pre-trained model with ability to identify the reasoning process, we propose a novel chain-centric pre-training method, which takes the advantage of the clear table structure and table-passage connections to construct large-scale pseudo reasoning paths, and reversely generate questions. CARP framework has following advantages. Firstly, the hybrid chain models the interaction between table and text, and reduces the redundant information. Secondly, it provides a guidance for QA, and better interpretability of the intermediate reasoning process. Lastly, both the training of the extraction model and the pre-training corpus construction require no human-annotated reasoning process, which alleviates the data sparsity problem and broadens the potential applications of the framework.

Experiments show that our system achieves the state-of-the-art result on a large-scale table-and-text open-domain question answering benchmark OTT-QA. Notably, the effectiveness of the chain-centric pre-training method is proved by the significant performance boost of the chain extraction model. Results show that incorporating the hybrid chain enhances the QA model, especially for the questions requiring more complicated reasoning process. We summarize following contributions:

1) We propose to model the intermediate reasoning process for question answering over table and text, with a fine-grained hybrid chain.

2) We propose a novel pre-training method, which captures the reasoning process by pre-training on a synthesized reasoning corpus consisting of large-scale cross-modality reasoning paths and corresponding questions.

3) Experiments show that our system achieves the state-of-the-art result and further analysis proves the effectiveness of utilizing the hybrid chain and the pre-training method.

2 Task Definition

In this paper, we study the task of question answering over table and text in a challenging open-domain setting, because the supported knowledge is not always provided in a realistic application. The task (Chen et al., 2020a) takes a question as the input, then requires the systems to first retrieve supported tables and passages, and then make inference over the retrieved knowledge to derive a free-formed answer as the output. The answer is a span from either the table cells or the passages. One of the core challenges of this task is that problem solving always requires complex reasoning process across table and text, considering the cross-modality interaction and table structure.

3 Framework: CARP

Fig. 2 shows the pipeline of our CARP framework, which has three main parts: (1) a retriever that retrieves tabular and textual knowledge with the given question (§ 3.5); (2) a chain extractor that extracts hybrid chain from the retrieved knowledge (§ 3.2). (3) a reader that answers questions with retrieved knowledge and the extracted hybrid chains (§ 3.4). We detailedly illustrate the hybrid chain (i.e., definition, extraction, pre-training, and application in QA), and briefly introduce the retriever.

3.1 Hybrid Chain Notation

Hybrid chain logically reveals the fine-grained reasoning process from question to the answer across table and text. We define the hybrid chain as a sequence of nodes extracted from a fine-grained heterogeneous graph $G$, whose nodes $V$ contain the question, cells in the table and sentences in the related passages. One example of the hybrid chain is shown in Fig. 1. Two nodes in the graph are connected by edges $E$ defined by two types of connections: structural connections and contextual connections. The former indicates that pairs of cells within a same row (e.g., edge $c$ in Fig. 1), or a cell to the a sentence in its linked passage (e.g., edge $b$), are structurally connected. The latter indicates that pairs of nodes with relevant context (i.e., entity/keyword co-occurrence) are contextually connected (e.g., edge $a$ indicates co-occurred keyword “COVID-19”). Specifically, we use off-the-shelf named entity recognition model (Peters et al., 2017) to extract entities, and extract noun phrase and numerical items as keywords from the node context. Moreover, a table cell and a passage is linked by the entity linker as described in § 3.5.
3.2 Hybrid Chain Extraction

Here we introduce how to extract hybrid chains, including the model architecture, training and inference process.

3.2.1 Model Architecture

We tackle the chain extraction as a semantic matching problem, which selects the best chain from several candidate chains. Taking a question and a candidate hybrid chain as the inputs, the model calculates the confidence score of the hybrid chain for answering the question. Each candidate hybrid chain is represented as a flattened sequence of its nodes context. Details and an example are given in the Appendix B.3. We utilize rich contextual representations embodied in pre-trained models like RoBERTa (Liu et al., 2019) to measure the relevance of a question to every chain candidates. Let’s take RoBERTa as an example. The input of the hybrid chain extractor is $input = ([CLS]; q; [SEP]; c_i)$ where $q$ and $c_i$ indicate tokenized word-pieces of the question and the flattened $i^{th}$ chain candidate. The $[SEP]$ and $[CLS]$ are special symbols. The representation $h_{c_i} \in \mathbb{R}^d$ is obtained via extracting the hidden vector of the $[CLS]$ token. The score $s^{+}_i$ for raking the candidates is calculated by:

$$s^{+}_i = \text{softmax}(Wh_{c_i} + b)$$

where $W$ and $b$ are the learnable parameters. The model is trained with the cross-entropy loss.

3.2.2 Model Training

As mentioned above, the key challenge is constructing the training instances (i.e., ground-truth chains and negative chains), as there is no gold-annotated reasoning process given as a prior.

We first introduce how to build ground-truth hybrid chains from the heterogeneous graph $G$. Partly inspired by Chen et al. (2019a), we use a heuristic algorithm to derive pseudo ground-truth hybrid chains. Starting from the question, we do the exhaustive search to find all the shortest paths to the nodes containing the answer as the candidate chains. Then, we select the best chain from all the candidate chains that have maximum textual similarity with the question as the final ground-truth hybrid chain, and take it as the positive instance. To build the hard negative instances, we find the shortest paths from the question node to the non-answer nodes and select the one with maximum textual similarity with the question.

3.2.3 Model Inference

During Inference, we first build a set of candidate hybrid chains from the graph $G$, and adopt the extraction model to rank all the chains, and finally select the best chain with highest confidence score.

More specifically, the set of whole candidate hybrid chains contains the shortest paths from the question node to all the other nodes in the graph. Suppose the number of nodes is $n$ in the graph, the number of candidate chains is $\sum_{i=0}^{n-1} SP(i)$, where $SP$ is the number of shortest paths to node $i$.

3.3 Chain-centric Pre-training

Pre-training for reasoning is always challenging because high-quality reasoning data is hard to be obtained. To better help the pre-trained model in capturing the complicated reasoning process across ta-
ble and text and alleviate the data sparsity problem, we propose a chain-centric pre-training method. The method augments the chain extraction model by pre-training on a synthesized reasoning corpus in larger scale and of higher reasoning complexity. The overall process of adopting pre-training strategy is illustrated in Fig. 3: (1) synthesizing heterogeneous chains from the Wikipedia corpus and reversely generating corresponding questions by a trained generator; (2) pre-training a generic extraction model with the synthesized corpus; (3) fine-tuning a specific extraction model with the downstream data. We introduce the pre-training task and the corpus construction.

**3.3.1 Task Formulation**

The pre-training task can be viewed as a similar semantic matching task that maps hybrid chains to the corresponding pseudo questions. The pre-training objective is in the same spirit of the chain extraction model as described in § 3.2. If the model can better distinguish the relevant hybrid chain for answering the given question, then it has deeper understanding of the reasoning process.

**3.3.2 Corpus Construction**

To construct the large-scale reasoning corpus, we adopt a novel way of first synthesizing heterogeneous reasoning paths, and then reversely generating corresponding questions. Tables in Wikipedia often contain hyperlinks to their related passages. The clear table structure and the explicit table-text links provide natural benefits for automatically synthesizing logically reasonable reasoning paths.

Therefore, we select semi-structured tables on Wikipedia as the table source, and take the passages hyper-linked to the table cells as the source of passages. The parsed Wikipedia corpus consists of over 200K tables and 3 million of hyperlinked passages. Then, we synthesize pseudo chains with different reasoning depths. For example, to synthesize a 4-hop reasoning path, we randomly select two cells \((c_0, c_1)\) within the same row and their related passages \((p_0, p_1)\) to form a chain \((p_0, c_0, c_1, p_1)\). Similarly, \((p_0, c_0)\) or \((c_0, c_1, p_1)\) can be selected as a 2-hop or a 3-hop chain, respectively.

Finally, taking a synthesized flattened chain as the input, we adopt a generation model built based on BART (Lewis et al., 2019) to reversely generate a pseudo question to construct a pair of \((question, chain)\) as a positive instance. It is worth noting that the generation model is trained by the ground-truth \((question, chain)\) pairs as described in § 3.2. To encourage the model to better discriminate relevant chains, we select other chains sampled from the same table with top-\(n\) similarity with the question as the hard negative instances.

**3.4 Hybrid Chain for QA**

Having extracted the hybrid chains for each table segment and its related passages, we need to build a reader model to extract the answer with the inputs. We build a reader model based on a sparse-attention based Transformer architecture Longformer (Beltagy et al., 2020) to process long sequence efficiently. With longer limited length up to 4096 tokens, the reader can read top-\(k\) retrieved evidences jointly for question answering. The input sequence \(x\) is the concatenation of the question and top-\(k\) pairs of \((table segment, passages, hybrid chain)\). The Longformer encodes the input \(x\) of length \(T\) into a sequence of hidden vectors:

\[
h(x) = [h(x)_1, h(x)_2, \ldots, h(x)_T]
\]

The probabilities \(p_{\text{start}}(i)\) and \(p_{\text{end}}(i)\) of the start and ending token of the answer \(a\) are calculated by:

\[
p_{\text{start}}(i) = \frac{\exp(W_s h(x)_i + b_s)}{\sum_j \exp(W_s h(x)_j + b_s)}
\]

\[
p_{\text{end}}(i) = \frac{\exp(W_e h(x)_i + b_e)}{\sum_j \exp(W_e h(x)_j + b_e)}
\]

where \(W_s, W_e, b_s, b_e\) are learnable weights and bias parameters of the answer extraction layer. Specifically, to alleviate the bias that the model only looks at the extracted chain, we only set the
chain as a guidance of the intermediate reasoning process and force the model to select answer from the tokens of the table and passages.

3.5 Knowledge Retrieval

Unlike retrievers in text-based open-domain QA systems, the retriever for this task is required to search both supported passages and tables. We briefly introduce the retriever in the last part for integrality, as it is not the main focus of our paper.

Instead of independently retrieving tables and passages, we follow Chen et al. (2020a) and use an “early-fusion” mechanism, which groups highly-relevant table cells in a row and their related passages as a self-contained group (fused block). This strategy integrates richer information from two modalities and benefits following retrieval process. We adopt BLINK (Ledell et al., 2020) as the entity linker to link a table cell to its related passage. BLINK is a highly effective BERT-based entity linking model and is able to link against all Wikipedia entities. Specifically, taking the cell to be linked and the table metadata as the inputs, BLINK automatically finds the relevant passages for each cell. After the linking procedure, we represent each fused block as a row in the table and linked related passages. Further details are given in the Appendix. We then tackle the fused block as a basic unit to be retrieved.

Finally, a Transformer-based retriever is employed to retrieve top-k fused blocks as the knowledge. We apply a shared RoBERTa-encoder RoBERTa(·) (Liu et al., 2019) to separately encode questions and fused blocks. The relevance of the question and a fused block is measured by the dot-product over their representations of the [CLS] token. We train the retriever model as in Karpukhin et al. (2020), where each question is paired with a positive fused block and m negative blocks to approximate the softmax over all blocks. Negative blocks are a combination of in-batch negatives which are fused blocks of the other instances in the mini-batch, and hard negative blocks which are sampled from the other rows in the same table. During inference, we apply the trained encoder to all fused blocks and index them with FAISS (Johnson et al., 2021) offline.

4 Experiments

In this section, we conduct experiments to explore the effectiveness of our method from the following aspects: (1) the performance of our overall system on QA; (2) the performance of the hybrid chain extraction model; (3) the ablation study about the pre-training strategy; (4) the comprehensive qualitative analysis. The retrieval performance and implementation details of all components are described in Appendix A and B, respectively.

4.1 Dataset and Evaluation

In the real-world scenario, solving many questions requires retrieving supporting heterogeneous knowledge and making reasoning over it. Therefore, we evaluate the performance of our approach on the OTT-QA (Chen et al., 2020a) dataset. OTT-QA is a large-scale table-and-text open-domain question answering benchmark for evaluating open-domain question answering answering over both tabular and textual knowledge. As the data statistics shown in Table 1, OTT-QA has over 40K instances and it also provides a corpus collected from Wikipedia with over 400K tables and 6 million passages. Furthermore, the problem solving in OTT-QA requires complex reasoning steps. The reasoning types can be divided into several categories: single hop questions (13%), two hop questions (57%), and multi-hop questions (30%). We adopt the exact match (EM) and F1 scores (Yu et al., 2018) to evaluate the overall QA performance.

<table>
<thead>
<tr>
<th>Type</th>
<th>Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Examples</td>
<td>41,469</td>
</tr>
<tr>
<td>Evaluating Examples</td>
<td>2,214</td>
</tr>
<tr>
<td>Testing Examples</td>
<td>2,158</td>
</tr>
<tr>
<td>Tables in the Corpus</td>
<td>410,740</td>
</tr>
<tr>
<td>Passages in the Corpus</td>
<td>6,342,314</td>
</tr>
</tbody>
</table>

Table 1: Data statistics of OTT-QA dataset.

4.2 Baselines

We compare our system to the following methods:

- **HYBRIDER** (Chen et al., 2020b) is a model that uses BM25 to retrieve relevant tables and passages, and adopts a two stage model to cope with heterogeneous information.

- **Iterative Retriever and Block Reader** The model family is proposed by Chen et al. (2020a), which couples Iterative Retriever (IR) / Fusion Retriever (FR) with Single Block Reader (SBR) / Cross Block Reader
To verify the effectiveness of our proposed hybrid chain, we firstly eliminate hybrid chain from the QA model inputs, and report the result of “CARP w/o hybrid chain” on the development set in Table 2. Incorporating hybrid chain into the QA model improves the performance significantly.

Then, we explore the performance of various variants in hybrid chain extraction, whose backbone is the pre-trained model RoBERTa (Liu et al., 2019). The variants consider three aspects: (1) encoding strategies; (2) ways of heterogeneous graph construction; (3) negative sampling strategies.

1. Dual Ranking vs Cross Matching: Dual-tower ranking model (Karpukhin et al., 2020) encodes the question and the hybrid chain separately, and uses the cosine-distance to measure their relevance for ranking. Cross matching means that we use a semantic matching model as described in § 3.2.

2. Simple (S) vs Weighted (W): Simple indicates the edges in the graph are unweighted. Weighted graph means that the edges connecting highly-related (higher ratio of overlapped keywords) nodes have lower weight, and thus the paths with higher overall relatedness (shorter length) are ranked higher in the ground-truth chain construction (§ 3.2).

3. BMNeg vs InnerNeg: BMNeg means that the most similar chain from other positive instances with BM25 are selected as the negative instance. InnerNeg indicates that we select negative instances from other chains constructed from the same fused block, as described in § 3.2.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Rec@1</th>
<th>Rec@2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dual Ranking (W + InnerNeg)</td>
<td>61.61</td>
<td>73.13</td>
</tr>
<tr>
<td>Cross Matching (W + BMNeg)</td>
<td>44.21</td>
<td>61.14</td>
</tr>
<tr>
<td>Cross Matching (S + InnerNeg)</td>
<td>68.32</td>
<td>79.87</td>
</tr>
<tr>
<td>Cross Matching (W + InnerNeg)</td>
<td>70.75</td>
<td>80.19</td>
</tr>
</tbody>
</table>

Table 3: Performance of the hybrid chain extraction model with different variances.
negative sampling strategy is extremely essential for hybrid chain selection. The goal of inference is to select the most plausible chain from several candidate chains sampled from the same fused block. Therefore, sampling hard negative instance from the same fused block is much better than sampling from other training instances. We take the setting of “Cross Matching (W + InnerNeg)” as the final setting of the extraction model.

4.5 Evaluation of Chain-centric Pre-training

In this part, we evaluate the effectiveness of the chain-centric pre-training strategy under different settings. The table cells are aligned to the passages according to their hyperlinks in the Wikipedia website. The main variance of pre-training is the different way of constructing instances for training the BART-based generator. All means that we take all the paths from the question node to the answer node as positive chains to train the generator. Shortest indicates that we only select the shortest paths.

As shown in Table 4, the pre-training strategy improves the performance of the hybrid chain extraction model by a large margin, showing the effectiveness of chain-centric pre-training in helping the model to capture the intermediate reasoning process with given questions. We believe that several reasons for the improvement of chain-centric pre-training are as follows. Automatically synthesizing pre-training data is an effective data augmentation scheme because it can generate data in larger scale and of higher reasoning complexity, which can help the model to better capture the complicated reasoning steps by pre-training.

Besides, selecting all paths leading to answer as positive chains to train the generator is better than selecting the shortest paths. This observation is intuitively reasonable since the goal of pre-training is to encourage the model to learn a more general reasoning ability with all possible reasoning paths.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Rec@1</th>
<th>Rec@2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extractor</td>
<td>70.75</td>
<td>80.19</td>
</tr>
<tr>
<td>Extractor + Pre-training (Shortest)</td>
<td>73.40</td>
<td>82.87</td>
</tr>
<tr>
<td>Extractor + Pre-training (All)</td>
<td>74.01</td>
<td>83.46</td>
</tr>
</tbody>
</table>

Table 4: Performance of the chain extraction with chain-centric pre-training under different settings.

4.6 Qualitative Analysis

We randomly select 100 instances from the development set and manually annotate the plausible hybrid chains and conduct qualitative analyses on several aspects: (1) the performance on the questions requiring different reasoning steps; (2) a case study by giving an example; (3) an analysis of common error types to shed light on future directions.

Performance on M-hop Questions

As shown in Fig. 4, we report the performance of the baseline (CARP without hybrid chain) and CARP on the selected questions with different reasoning steps. It can be observed that as the number of reasoning steps increases, the improvement brought by our method to the baseline becomes more significant. This observation verifies that, the hybrid chain is essential in helping the model to identify the intermediate reasoning steps towards the answer especially when the reasoning is more complicated. Our synthesized pre-training corpus includes higher ratio of 3-hop questions, which enhance the multi-hop reasoning ability of the system.

Case Study

We conduct a case study by giving an example shown in Fig. 5. From the example, our chain extraction model selects a semantic-consistent hybrid chain from the fused block and the QA model correctly predicts the answer with the help of the hybrid chain. This observation reflects that our model has the ability to extract intermediate reasoning process from the given inputs and utilize this information to facilitate the question answering process. Hybrid chain also makes the predictions become more interpretable.

Error Analysis

We summarize major types of errors to shed light on future directions. The most common type of errors is caused by the disturbance of wrongly retrieved fused blocks because we feed...
The squad currently plays at Argentine Primera Division. The localidad is home to Chacarita Juniors, a football club that won the 1969 Metropolitano.

Figure 5: A case study of our approach. The answer is Argentine Primera Division. We omit some unimportant sentences in the passage for simplification.

Our approach differs from previous methods mainly in two aspects: (1) our method formulate heterogeneous chain to model the complex reasoning process across table and text; (2) the chain-centric pre-training method can enhance reasoning ability of models by pre-training on a synthesized reasoning corpus, containing heterogeneous reasoning paths and pseudo multi-hop questions.

6 Conclusion

In this paper, we present a chain-centric reasoning and pre-training (CARP) framework for table-and-text question answering. When answering the questions given retrieved table and passages, CARP first extracts explicit hybrid chain to reveal the intermediate reasoning process leading to the answer across table and text. The hybrid chain provides a guidance for QA, and explanation of the intermediate reasoning process. To enhance the extraction model with better reasoning ability and alleviate data sparsity problem, we design a novel chain-centric pre-training method. This method synthesizes the reasoning corpus in a larger scale and of higher reasoning complexity, which is achieved by automatically synthesizing heterogeneous reasoning paths from tables and passages in Wikipedia and reversely generating multi-hop questions. We find that the pre-training task boosts performance on the hybrid chain extraction model, especially for questions requiring more complex reasoning, which leads to significant improvement on the performance of the QA model. The hybrid chain also provides better interpretability of the reasoning process. Our system achieves the state-of-the-art result on a table-and-text open-domain QA benchmark.
References


A Evaluation of Retrieval Model

In this part, we evaluate the retrieval performance of retrievers.

A.1 Settings

Our retriever is evaluated on the OTT-QA dataset (Chen et al., 2020a), which is a large-scale open-domain question answering dataset over table and text. We compare our retriever with the following retrieval methods. (1) BM25 (Chen et al., 2020a) is a sparse method to retrieve tabular evidence with BM25. It represents the table with the flattened sequence of table metadata (i.e., table title and section title) and table content. (2) Bi-Encoder (Kosti’c et al., 2021) is a dense retriever which uses a BERT-based encoder for questions, and a shared BERT-based encoder to separately encode tables and text as representations for retrieval. (3) Tri-Encoder (Kosti’c et al., 2021) is a dense retriever that uses three individual BERT-based encoders to separately encode questions, tables, and text as representations.

A.2 Evaluation Metrics

In this experiment, we use two metrics to evaluate the retriever: table recall and fused block recall. Table recall indicates whether the top-\(k\) retrieved blocks come from the ground-truth table, which is also used in other papers. However, in tablertext retrieval, table recall is imperfect as a coarse-grained metric since our basic retrieval unit is a table-text block. Therefore, we use a more fine-grained and challenging metric: fused block recall at top-\(k\) ranks, where a fused block is considered a correct match when it meets two requirements: coming from the ground truth table and containing the correct answer.

A.3 Performance

The results are shown in Table 5. We can find that our retriever substantially outperforms sparse BM25 method and achieves comparable performance with Bi-Encoder and Tri-Encoder.

<table>
<thead>
<tr>
<th>Models</th>
<th>Table Recall</th>
<th>Block Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R@1 R@10 R@100</td>
<td>R@1 R@10 R@100</td>
</tr>
<tr>
<td>BM25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bi-Encoder</td>
<td>- 72.9 89.4</td>
<td>- - -</td>
</tr>
<tr>
<td>Tri-Encoder</td>
<td>- 73.8 90.1</td>
<td>- - -</td>
</tr>
<tr>
<td>Ours</td>
<td>49.0 74.0 88.6</td>
<td>16.3 46.7 75.5</td>
</tr>
</tbody>
</table>

Table 5: Overall retrieval results on OTT-QA dev set. Table recalls and fused block recalls are reported.

B Implementation Details

B.1 Fused Block Representation

In this part, we describe how we represent a fused block with a table row and its related passages. Similar to Chen et al. (2020a), we represent each fused block as the concatenation of the table meta data, the cells in the rows, and related passages: \(Fused\ Block = \left([TAB] [TITLE] title [DATA] row [PASSENGES] passages\right)\), where \(row\) and \(passages\) indicate the flattened row and all the related passages of this row, and there is a \([SEP]\) token between cells or passages.

B.2 Retrieval Model

In this part, we describe the details of the fused block retrieval model. Our retrieval model follows a typical dual-encoder architecture, which uses a dense encoder \(E(\cdot)\) to map any fused block to a \(d\)-dimensional dense vector and build an index for all the blocks for retrieval. At query time, the input question \(q\) is mapped to a \(d\)-dimensional dense vector by the same neural encoder \(E(\cdot)\), and returns top-\(k\) fused blocks that are closest to the question representation. The similarity of \(q\) and \(b\) is measured by a dot-product of two vectors:

\[
sim(q, b) = E(q)^\top \cdot E(b). \tag{4}
\]

In practice, we use a pre-trained RoBERTa-base (Liu et al., 2019) to initialize our encoder and take the representation at the first token (i.e. \([CLS]\) token) as the output. At inference time, we apply FAISS (Johnson et al., 2021) to index the dense representations of all fused blocks.

B.2.1 Training

The training objective aims to maximize the probability of positive pairs. Formally, given a question \(q_i\) together with its positive block \(b_i^+\) and \(m\) negative blocks \(\{b_{i,1}^-, \ldots, b_{i,m}^-\}\), we optimize the loss function as the negative log-likelihood of positive
Following Karpukhin et al. (2020), we use 1 hard negative fused block randomly sampled from the same table, and \( m − 1 \) in-batch negatives during training.

**B.3 Hybrid Chain Extraction Model**

In this part, we describe the example of the flattened hybrid chain and training details of our hybrid chain extraction model.

**Verbalization of the hybrid chain** We introduce how to represent hybrid chain with natural language, and enable the powerful pre-trained language model to calculate its contextual representations. Each node is either the question, a table cell or a sentence in the passages. Therefore, we represent the content in different types of nodes as: "[Question] (question)", "[Table] (column_name is (cell_content))" or "[Passage] (sentence)", respectively. [Question], [Table], [Passage] denote special symbols. Then, we concatenate the context in all the nodes corresponding to their types, and separate them with a "[SEP]" special symbol. In our experiment, we omit the question node from the final sequence, to avoid exceeding the maximum sequence length limit of the pre-trained models.

For example, the hybrid chain in Fig. 1 can be represented as: "[Question] How many ... COVID-19? [SEP] [Table] The season ... COVID-19. [SEP] [Table] Year is 19-20. [SEP] [Table] Points is 25.3."

**Training Details** We employ cross-entropy loss as the loss function. We apply AdamW as the optimizer for model training. We employ RoBERTa-Base as the backbone of our approach. We set the learning rate as 1e-5, warmup step as 0, batch size as 16 per GPU, and set max sequence length as 512. The training time for one epoch takes 1 hours on 8 V100 GPUs.

**B.4 Chain-centric Pre-training**

**Corpus Construction** When constructing the pre-training corpus, we use 3 millions pairs of (question, hybrid chain) as the positive training instances, and search for the same number of hard negative instances, and the final pre-training corpus contains nearly 6 millions of training instances. It worth noted that, to avoid the bias caused by the length of the hybrid chain, we automatically synthesize hybrid chains with different length various from 1 to 4. The ratio of the synthesized chains with different lengths are: 1-hop (0.1); 2-hop (0.25); 3-hop (0.35); 4-hop (0.3). As for the pseudo questions generator, we employ BART-Large as the backbone. It is firstly trained upon pairs of our extracted hybrid chains and questions from the OTT-QA dataset. During training, its learning rate is set as 3e-5, warmup step is as 2000, and batch size is as 8 per GPU. The training time for one epoch takes nearly 2 hours on 8 V100 GPUs.

**B.5 QA Model**

We employ the Longformer-Base (Beltagy et al., 2020) as the backbone of our QA model. We set batch size as 2 per GPU, set max sequence length as 512, and set document stride as 3072. The learning rate is 1e-5. The training time for one epoch takes 3 hours on 8 V100 GPUs. We concatenate top-15 fused block as the evidence for both training and inference. We adopt AdamW as the optimizer, and use cross entropy as the loss function. During training and inference, we force the model to only select the answer from the tokens of the fused blocks.