# Table Retrieval Does Not Necessitate Table-specific Model Design

**Anonymous ACL submission**

## Abstract

Tables are an important form of structured data for both human and machine readers alike, providing answers to questions that cannot, or cannot easily, be found in texts. Recent work designs special models and trains for table-related tasks such as table-based question answering and table retrieval. Though effective, they add model-data dual complexity to generic text solutions and obscure which elements are truly beneficial. In this work, we focus on the task of table retrieval, and ask: "are table-specific model designs necessary for table retrieval, or can a text-generic model be effectively used to achieve a similar result?" We start by analyzing NQ-table, a set of table-answerable questions in the Natural Questions (NQ) dataset, and find 90% of the questions can match tables in content with little concern for table structure. Motivated by this, we experiment with a general-purpose Dense Passage Retriever (DPR) for text and a special-purpose Dense Table Retriever (DTR) for tables. We show that DPR, without any design for or training on tables, can perform comparably well to the state-of-the-art DTR model, and neither adding DTR-like table-specific embeddings nor perturbing cell orders lead to significant changes. Both results strongly indicate that table retrieval does not necessitate table-specific model design, as well as the potential of directly applying powerful text-generic retrievers to structured tables.[1]

## 1 Introduction

Tables are a valuable form of data that organize and distill information in a structured way for easy storage, browsing, and retrieval (Cafarella et al., 2008; Jauhar et al., 2016; Zhang and Balog, 2020). They often contain data that is not available in text (Chen et al., 2020a), or data records that are organized in a more accessible manner than in unstructured texts. Therefore, tables are widely used in Question Answering (QA) (Pasupat and Liang, 2015; Zhong et al., 2017; Yu et al., 2018). For open-domain QA, the ability to retrieve relevant tables with target answers is crucial to the performance of end-to-end QA systems (Herzig et al., 2021). For example, in the Natural Questions (Kwiatkowski et al., 2019) dataset, 13.2% of the answerable questions can be addressed by tables.

**Question**: Who won the most Stanley Cups in history?

Stanley Cup Finals

From Wikipedia, the free encyclopedia

**Most Finals appearances (top five)**

**(Bold indicates Cup wins)**

| Appearances | Team | Wins | Losses | Win % | Years of appearance |
|---|---|---|---|---|---|
| 35[3] | Montreal Canadiens (NHA/NHL) | 24 | 10 | .706 | **1916**, 1917, 1919[3], **1924**, 1925, **1930**, **1931**, **1944**, **1946**, 1947, 1951, 1952, **1953**, 1954, 1955, **1956**, **1957**, **1958**, **1959**, **1960**, **1965**, **1966**, 1967, **1968**, **1969**, **1971**, **1973**, **1976**, **1977**, **1978**, **1979**, **1986**, 1989, **1993**, 2021 |
| 24 | Detroit Red Wings | 11 | 13 | .458 | 1934, **1936**, **1937**, 1941, 1942, **1943**, 1945, 1948, 1949, **1950**, **1952**, **1954**, **1955**, 1956, 1961, 1963, 1964, 1966, 1995, **1997**, **1998**, **2002**, **2008**, 2009 |

Figure 1: A correct table can be identified by matching key phrases in question to those in the table title and header cells.

Because tables are intuitively different from unstructured text, most previous work has considered text-based methods to be functionally incapable of processing tables effectively (Herzig et al., 2020; Yin et al., 2020; Wang et al., 2021b; Liu et al., 2021). Recent work has created special-purpose models for table-related tasks with structure encoding modules: inserting additional parameters in the embedding (Herzig et al., 2020; Wang et al., 2021b; Deng et al., 2020) or attention (Yin et al., 2020; Wang et al., 2021b; Zayats et al., 2021) layers, then deliberately pre-training models using table-oriented objectives (Deng et al., 2020; Yin et al., 2020; Wang et al., 2021b; Liu et al., 2021; Yu et al., 2020). Though effective in many tasks, these special-purpose models are more complex than generic solutions for textual encoding, and must be intentionally built for and trained on tabu-

[1] Code and data to reproduce the experiments will be released upon acceptance.

lar data. In addition, because these methods modify both the model design and the training data, it is difficult to measure the respective contributions of each of these elements. Particularly for question-based table retrieval, which emphasizes content more than table structure, we argue that the benefit may well come from good training data while the model architecture has a limited influence. For example, given a question "Who won the most Stanley Cups in history?" in Fig. 1, a correct table can be retrieved by simply identifying the topic "Stanley Cup" in the table title and the words "Wins" and "Team" among header cells.

In this paper, we focus on the task of table retrieval and ask: "Does table retrieval require special-purpose representations, or can properly trained text-based models be exploited to achieve similar results with less added complexity?" Our work centers around the table-based open domain question answering dataset, NQ-table (Herzig et al., 2021), a subset of the NaturalQuestions dataset (Kwiatkowski et al., 2019) where each question can be answered by part(s) of a Wikipedia table.

We start with a manual analysis of question-table matching patterns using 100 random NQ-table test samples. This analysis reveals that 90% of the pairs can be identified by table content with little structural information, moreover, 75% involves phrases in table title and header cells only (§ 2) With this insight, we further experiment with two strong models for retrieval: a general-purpose text-based retriever (DPR; Karpukhin et al. (2020)) and a special-purpose table-based retriever (DTR; Herzig et al. (2021)). We demonstrate that DPR, benefiting from training on large textual data, can retrieve tables comparably well with the state-of-the-art table retriever DTR, which is specifically designed for and heavily trained on tables (§ 3). We further study the potential benefit of injecting table-specific inductive bias into DPR via augmenting its training data with tables and table-specific model modifications. Our experiments show that training on the target NQ-table dataset leads to a significant increase in retrieval accuracy, but explicit embedding layers for tabular features do not (§ 4). In sum, the results reveal that a strong text-based model is competitive for table retrieval, indicating the potential to directly apply future improved text retrieval systems for retrieving structured tables, a task they were previously considered less applicable.

## 2 Question-table Matching Analysis

To better understand the requirements inherent in question-based table retrieval, we start with a detailed manual analysis of NQ-table dataset. We randomly sample 100 annotated question-table pairs to see which parts of the table can allow for matching to the questions, which aspects do they specify, and in what combinations are they utilized. Our analysis identifies three major matching patterns, which are further verified in experiments in § 5.1.

### 2.1 The NQ-table Dataset

The NQ-table dataset has been recently collected by Herzig et al. (2021) from the Natural Questions (NQ) dataset (Kwiatkowski et al., 2019). The original NQ dataset contains scraped search queries that can be answered by Wikipedia pages. We focus on the portion where the annotated answers are short spans in relevant pages. Previous work only keeps the text in the source HTML pages that can answer around 59k questions. In addition, 12k questions are answerable by content in Wikipedia tables.

### 2.2 Matching Questions with Tables: Title, Header, and Content

Questions in NQ(-table) usually ask about a certain aspect (e.g. who, when) of a main topic, often with one or more descriptive details. For example the question in Fig. 1 asks "who" about the topic entity "Stanley Cup" and specifies "won" "the most". Through this analysis, we observe the alignment of phrases between questions and tables. Indicative phrases about entities, properties, and semantic types in questions often match those in tables, especially in their title and header regions. Based on this observation, we break down tables into three components — title, header, and content — to study "based on which table components" and "through what mechanisms" can matching be performed.

**Table Title Directly Matches the Question** For 18 out of the 100 sampled questions, we can confidently match each with the annotated table by its title, given its informative and indicative wording.

**Question**: Who is the highest paid baseball player in the major leagues?

**Table**:

List of highest-paid Major League Baseball players

| Name | Position | Team(s) | Salary | Ref |
|---|---|---|---|---|
| Clayton Kershaw | SP | Los Angeles Dodgers | $32,571,428 | [15] |
| Justin Verlander | SP | Houston Astros | $28,000,000 | [16] |
| Ryan Howard | 1B | Philadelphia Phillies | $25,000,000 | [17] |
| Cliff Lee | SP | Philadelphia Phillies | $25,000,000 | [18] |

Figure 2: A table that matches the question by its title.

For the example in Fig. 2, the table title is in good detail, including both the topic "baseball player" and the specifications "highest paid" "major league". Moreover, the "List of" "players" are highly indicative of displaying a name list to answer the type "who".

**Header Cells Imply the Answer Type** In another 57 samples, one can only infer the relevance of the matched topic using the title, but whether the table contains items of the requested type (e.g. who) and fit the specification remains unknown. Header cells can indicate the semantic types of table columns, hence, the relevance of table can be decided based on whether they match to the type asked for in the question. As shown in Fig. 3, the concise title "Bald Eagle" matches only the topic; the header cell "Genus" further matches the asked entity and confirms the answer to "What".

**Question**: What is the genus of a bald eagle?

**Table**:

| Bald eagle | |
|---|---|
| Kingdom: | Animalia |
| Phylum: | Chordata |
| ..... | |
| Family: | Accipitridae |
| Genus: | *Haliaeetus* |
| Species: | ***H. leucocephalus*** |

Figure 3: A sample of matching topic in table title and answer type in table header.

Concluding from the two patterns so far, about 75% of the matching finds the combination of title and header sufficient, suggesting that the topical information is largely condensed into the title and header regions.

**Content Cells Clarify Table Semantics** Another 15 examples involve unclear headers, which have either general wording or only a distant relation to the specific entities being asked, therefore, requiring further confirmation of the table content cells. For the case in Fig. 4, besides the "playoff" and "time" matching, we need to scrutinize the table content for "jaguars" to confirm its mention in the "franchise" column. Inferring or confirming a target mention is critical to judge the relevance of a table.

Up until now, the three patterns add up to 90%. This large portion of all questions concerns only the table content with little necessity to reference structural information, indicating little necessity for consideration of structure structure, the main component of table-specific model designs.

**Question**: When is the last time the jaguars won a playoff game?

**Table**: List of NFL franchise post-season droughts

| Franchise | Most recent division title | Year | Seasons |
|---|---|---|---|
| ..... | | | |
| Cincinnati Bengals | AFC North | 2015 | 6 |
| Denver Broncos | AFC West | 2015 | 6 |
| Atlanta Falcons | NFC South | 2016 | 5 |
| Jacksonville Jaguars | AFC South | 2017 | 4 |
| Minnesota Vikings | NFC North | 2017 | 4 |

Figure 4: A table matches the question with additional indication of semantic type in the content, beyond topic in title and type indication in header cells.

**Use Structure to Check Answer** Although few, 3 cases find content alone is insufficient and require table structure. For the example in Fig. 5(a), under the general header "Population", one should locate the sub-header "Total" via their structural correspondence to confirm that the 'total number' measure of the 'population' topic exists.

**Question**: What is the population of Keystone Heights, Florida?

**Table**:

| Keystone Heights, Florida | |
|---|---|
| ..... | |
| Country | United States |
| State | Florida |
| County | Clay |
| Area[1] | |
| · Total | 1.09 sq mi (2.82 km²) |
| · Land | 1.07 sq mi (2.78 km²) |
| · Water | 0.01 sq mi (0.04 km²) |
| Elevation | 141 ft (43 m) |
| Population (2020) | |
| · Total | 1,446 |
| · Density | 1,345.12/sq mi (519.52/km²) |

(a)

**Question**: How many scholarships do Division 2 football teams get?

**Table**: NCAA Division II

| Sport | Men's | Women's |
|---|---|---|
| Acrobatics & tumbling | – | 9.0 |
| Baseball | 9.0 | – |
| Basketball | 10.0 | 10.0 |
| ..... | | |
| Fencing | 4.5 | 4.5 |
| Field hockey | – | 6.3 |
| Football | 36.0 | – |
| Golf | 3.6 | 5.4 |
| Gymnastics | 5.4 | 6.0 |

(b)

Figure 5: (a) requires both table content and its structure to check for answer. In (b) it is hard to match the table with the question, due to the ambiguous question and lack of table description.

**Other Hard-to-Answer Questions** Besides the above three categories, there are 7 out of 100 samples that we fail to find a confident match between the question and the ground truth table. Two major reasons make these questions hard to answer. First, similar to the finding of Min et al. (2020), the question is expressed ambiguously. Second, descriptions about table content may be insufficient. The case in Fig. 5(b) exemplifies both reasons. First, while the question only specifies the "football team", the table documents "Men" and "Women" respectively. It is unable to determine which gender that is being asked for. Second, nowhere in the table content mentions the "scholarship" being asked about. Without further descriptions in the table content, it is difficult to relate the "Men's" and "Women's" values to "scholarship".

## 3 Text and Table Retrievers

Given the observations from the previous dataset analysis, table content plays a much more important role in retrieval than table structure. We hence hypothesize that general-purpose text-based retrievers might not be necessarily worse than special-purpose table-based retrievers, contradictory to what most previous work has assumed (Herzig et al., 2021, 2020; Yin et al., 2020; Wang et al., 2021b). To assess our hypothesis, we examine two representative retrieval systems under the same experimental setting: the text-based Dense Passage Retriever (DPR) and the table-based Dense Table Retriever (DTR).

### 3.1 Text Retriever: DPR

We choose DPR (Karpukhin et al., 2020) as a representative text-based retrieval model, mainly because of (1) its impressive performance across many text-related retrieval tasks, and (2) its similarity with DTR from both training and modeling perspectives, which make it easy to make fair comparisons.

DPR comprises a question-context bi-encoder built on BERT (Devlin et al., 2018), which includes three types of input embeddings as summarized in Tab. 1. The question encoder $\text{BERT}_q$ encodes each question $\boldsymbol{q}$ and outputs its dense representation using the representation of [CLS] token, denoted as $\boldsymbol{h_q} = \text{BERT}_q(\boldsymbol{q})[\text{CLS}]$. The context encoder works similarly. To enable tables for sequential context inputs, we linearize each table into a token sequence $\boldsymbol{T}$, which is then fed into the context encoder $\text{BERT}_c$ to obtain its dense representation $\boldsymbol{h}_T = \text{BERT}_c(T)[\text{CLS}]$. Similarity score between a question $\boldsymbol{q}$ and a table $\boldsymbol{T}$ is computed as the dot product of two vectors $sim(\boldsymbol{q}, T) = \boldsymbol{h_q} \cdot \boldsymbol{h}_T$.

DPR has been trained only on text contexts within 100 words. For each question in the NQ-text training set, the model is trained to select the correct context that contains the answer from a curated batch of contexts including both the correct and mined hard negative contexts. Unless otherwise specified, we use NQ-text to denote the commonly referred NQ dataset that can be answered by texts.

To convert tables into the DPR input format, we linearize tables into token sequences. We concatenate the title, the header row, and subsequent content rows using a period '.' (row separator). Within each header or content row, we concatenate adjacent cell strings using a vertical bar '|' (cell separator). A template table linearization reads as the order $[\text{title}].[\text{header}].[\text{content}_1]. \cdots .[\text{content}_n]$. Although BERT encoder has the capacity for at most 512 tokens, DPR is only exposed to contexts no longer than 100 words during training and testing. To avoid potential discrepancies between its original training and our inference procedure, we shorten long tables by randomly selecting content rows to approximate the 100-word window [2].

| Embeddings | DPR | DTR |
|---|---|---|
| **token** | BERT vocab | BERT vocab |
| **segment** | 0 for all tokens | 0 for text, 1 for table |
| **position** | sequential | cell-wise reset |
| **row** | - | row index |
| **column** | - | column index |
| **rank** | - | rank of token value |

Table 1: Comparison of DPR and DTR embeddings.

### 3.2 Table Retriever: DTR

Dense Table Retriever (DTR) (Herzig et al., 2021) is the current state-of-the-art table retrieval model on the NQ-table dataset.

From the perspective of *model architecture*, DTR largely follows the bi-encoder structure of DPR, but differs from it in the embedding layer. DTR utilizes the existing embeddings in a slightly different way and introduces other types of embeddings specifically designed to encode table data. Tab. 1 shows a detailed comparison of their embeddings. Both models share the same BERT vocabulary index for token embedding. For the segment index, DPR assigns all tokens in a sequence to the same index 0, while DTR distinguishes the title from the table content by assigning 0 and 1, respectively. But for positional index, DPR inherits from BERT the sequence-wise order index $[0, 1, 2, ..., \text{sequence length} - 1]$, while DTR adopts a cell-wise reset strategy that records the index of a token within its located cell $[0, 1, ..., \text{cell length} - 1]$. In addition, DTR introduces row and column embeddings to encode the structural position of each token in the cell that it appears. This explicit joint of three positional embeddings is potentially more powerful than the BERT-style flat index. Furthermore, concerning the high frequency of numerical values in tabular data, DTR adds a ranking index for each token if it is part of a numerical value.

[2] Final length of the linearized tables is on average 113 words. A tiny number of extra words are allowed to compensate for the extra spaces taken by cell and row separators.

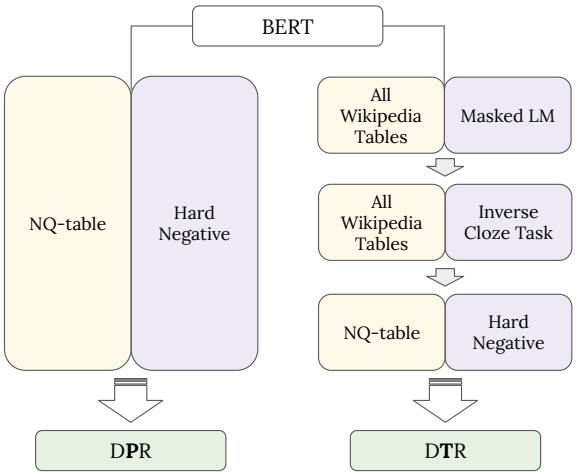


Figure 6: Comparison of DPR and DTR training.

Concerning the *training process* summarized in Fig. 6, DTR performs three stages of training on tables. First, the model parameters, except for those newly added table-specific embeddings, are initialized with pre-trained BERT weights. The model is then pre-trained on all Wikipedia tables using the Masked LM (MLM) task. The resulting model after this step is often referred to as TAPAS (Herzig et al., 2020). Second, to leverage TAPAS in the retrieval setting, it is further pre-trained using the Inverse Cloze task (ICT) introduced by ORQA (Lee et al., 2019), again, on all Wikipedia tables. Third, the model trains on the specific NQ-table dataset, similar to the way that DPR is trained on text retrieval datasets. For each question in the NQ-table training set, DTR uses the annotated table as the positive context and self-mined tables without answers as hard negative (HN) contexts.

### 3.3 Table Retrieval: Results and Analysis

We start with measuring the retrieval accuracy using the original DPR and DTR model without supplemental training. We load the published checkpoints of both models and evaluate on NQ-table test set, then measure the retrieval accuracy by assessing whether the retrieved table contains the answer. The published DPR and DTR checkpoints are not directly comparable, since the DPR "base" size, which falls between the DTR "medium" and "large" sizes with respect to the number of parameters. Hence, we measure DTR in both medium and large sizes as approximate lower and upper bounds for a "base" DTR model. More detailed model configurations for different sizes are listed in Appendix A.

The top-k retrieval accuracy of DTR and DPR is listed in Tab. 2. DPR is able to achieve an accuracy comparable to DTR on NQ-table without any table-specific model design or training. Even more impressively, starting from the top-3 metric, DPR-base readily outperforms DTR-large. DTR incorporates table-specific embeddings and trains heavily on tables from the entirety of Wikipedia as well as those in the NQ-table dataset. In contrast, DPR involves no table-specific modules and is only trained on text. The similarity in performance and differences in modeling and training align with our previous analysis that table retrieval may not necessitate table-specific model design.

| Model Name | Model Size | Retrieval Accuracy @1 | @3 | @5 | @10 | @20 | @50 |
|---|---|---|---|---|---|---|---|
| DTR | medium | 62.32 | 78.16 | 82.51 | 86.75 | 91.51 | 94.26 |
| DTR | large | **63.98** | 78.99 | 84.27 | 89.65 | 93.48 | 95.65 |
| DPR | base | 61.38 | **80.54** | **85.51** | **91.41** | **94.62** | **96.69** |

Table 2: Top-k table retrieval accuracy on NQ-table test.

Many differences do DPR and DTR have in embedding and training. While it is yet unclear to state the effect of embedding, which we will examine in § 4, we conjecture that training on text retrieval datasets can also benefit table retrieval. Although the context (text in DPR versus tables in DTR) could bring in potential gaps, the questions in NQ-text and NQ-table share similar characteristics and can be agnostic to answer sources (Wolfson et al., 2020). As such, a good NQ-text question encoder can be similarly used to represent NQ-table questions, especially when it is trained on a larger set of NQ-text questions (58,880) than on NQ-table (9,596). Motivated by this, we compare the questions in NQ-text and NQ-table to measure their extent of overlap. After basic NLTK (Bird et al., 2009) text normalization[3], we find that around 59.6% of questions in the NQ-table training set also appear in the NQ-text training set. This is rather intuitive since a question answerable by tables can also have answer spans in text, especially when the text and table describe the same subject on the same Wikipedia page.

In summary, our analysis implies that text retrieval and table retrieval are not two isolated tasks that are meant to be solved individually and specifically. As we will examine in § 4, training on both datasets is indeed a better choice.

[3]This is part of the standard normalization process of DPR.

## 4 Learning to Represent Tables

In this section, we study the potential benefit of learning table-specific representations based on the DPR model. First, we train DPR on the NQ-table training set to learn the patterns in table data, and examine "would simply training on table data improve DPR's performance?". Second, we train with additional table-oriented embeddings in the DPR context encoder, and examine "would introducing table-specific embeddings further help?".

### 4.1 Training on the NQ-table Dataset

To create training samples for DPR, we follow its curation strategy on NQ-text, and similarly create the NQ-table training samples using the annotated-positive and mined hard-negative tables. For each question in the NQ-table training set, we take the same positive table used in DTR training. For negative samples, we use the original DPR checkpoint to retrieve the top-100 table candidates for each question, among which we take the highest-ranked tables without the answer as the hard negatives.

Since the questions in NQ-text and NQ-table are largely the same, the distinction between textual and tabular context can generate more notable updates in the DPR context encoder. To prevent the updates in the context encoder from overly disturbing the question encoder, we first fix the question encoder and train the context encoder for one epoch. Then, we optimize both encoders and train for another epoch. Both experiments use a batch size of 16 and a learning rate of $2e{-}5$. More experimental details are described in Appendix B.

As shown by the **DPR-t** in Tab. 3, this two-epoch training drastically increases the model performance on the NQ-table test questions, especially in top-k metrics with smaller k. Since DPR has not been trained on table data, training on NQ-table samples helps it learn to consume tables and adapt from question-text to question-table mapping.

### 4.2 Injecting Table-Specific Embeddings into Context Encoder

Auxiliary training on the NQ-table dataset brings about a significant improvement. To examine if an explicit incorporation of table-specific features would further help, we augment the DPR context encoder with extra embeddings as those used in DTR (described in § 3.2). We substitute the DPR context encoder from BERT with a TAPAS encoder. Except for the newly added table-specific embeddings, other parts of the model are kept the same and initialized with the original DPR parameters. We initialize new embeddings to zero to prevent unwanted disturbance to the holistically learned DPR weights. Besides the weights, the modified DPR is identical to DTR in terms of model architecture.

The adjusted model is trained in three steps. We first freeze all modules except the newly introduced table-specific embeddings and train on NQ-table for one epoch. Next, we fix the question encoder and update the entire context encoder for one epoch. Lastly, we unfreeze all model parameters and train both encoders for one epoch. We use the same hyper-parameter setting to the previous experiment (batch size $= 16$ and learning rate $= 2e{-}5$). More details can be referred to in Appendix B.

| Model Name | Model Size | Retrieval Accuracy @1 | @3 | @5 | @10 | @20 | @50 |
|---|---|---|---|---|---|---|---|
| DTR | medium | 62.32 | 78.16 | 82.51 | 86.75 | 91.51 | 94.26 |
| DTR | large | 63.98 | 78.99 | 84.27 | 89.65 | 93.48 | 95.65 |
| DPR | base | 61.38 | 80.54 | 85.51 | 91.41 | **94.62** | 96.69 |
| DPR-t | base | **70.08** | **85.51** | 88.82 | 91.72 | 94.51 | **97.20** |
| DPR-e | base | 66.01 | 85.20 | **89.95** | **91.93** | 94.31 | 95.85 |

Table 3: Top-k table retrieval accuracy on NQ-table test. DPR is the original model checkpoint. **DPR-t** is tuned on the NQ-table training set. **DPR-e** is the embedding-augmented DPR tuned on the NQ-table training set.

As shown by the **DPR-e** results in Tab. 3, encoding tables with specific embeddings do not bring substantial improvement on the retrieval accuracy. Instead, the results with and without table-specific embeddings are comparable to each other across multiple top-k metrics. Nonetheless, we recognize that it can be hard to make the original DPR weights and new embeddings compatible in as few as three epochs, which could potentially result in the $-4.07$ drop in top-1 retrieval accuracy.

## 5 Ablation: Table Components and Their Structured Order

In this section, we conduct ablation studies in two aspects. For one, to experimentally corroborate our analysis on the NQ-table dataset in § 2, we adjust the inputs to DPR by using different combinations of table components and explore alternative table linearization methods. For another, given the minor significance of the structure demonstrated in § 4.2, we perform perturbations on the ordered table cell mapping to further consolidate this finding.

### 5.1 Table Components for DPR Inputs

To corroborate our analysis in § 2 and examine the utility of different table components (title, header, and content), we experiment with using different combinations of them to generate DPR inputs.

Tables on the same Wikipedia page have the same title, hence only using the title hardly distinguishes them and results poorly in our preliminary experiments. So, we start by examining the joint usage of title and table headers for input. Our first experiment linearizes each table by concatenating its title and header, which is denoted by **no-content** in Tab. 4. Because fewer table components are used, the resulting sequences can be shorter than the 100-word passages used at training. To mitigate this potential discrepancy, we follow Herzig et al. (2021) to repeat the resulting text for at most 15 times until they reach the 100 words, which is the default length limit of DPR model inputs.

Next, to study the additional benefit brought by table content, our second setting appends linearized table content rows into the input sequence. We explore two linearization methods. The first one uses the same input as the DTR model: table content is linearized in turns of rows, and, to maximize the number of cells that fits in a 512-token window, cell strings are truncated to a dynamic length threshold. We denote this by **all-cell-trunc.** in Tab. 4.

| Content | Retrieval Accuracy | | | | | |
|---|---|---|---|---|---|---|
| | @1 | @3 | @5 | @10 | @20 | @50 |
| no-content | 38.82 | 57.45 | 65.73 | 75.98 | 82.50 | 89.34 |
| all-cell-trunc. | 56.52 | 77.02 | 82.82 | 89.23 | 93.17 | 95.76 |
| all-row-trunc. | 58.28 | 78.26 | 83.13 | 88.51 | 92.86 | **96.69** |
| partial | **61.38** | **80.54** | **85.51** | **91.41** | **94.62** | **96.69** |

Table 4: Top-k table retrieval accuracy on NQ-table test set with various table components and linearizations. **no-content** refers to the first title-header setting with no content. **all-cell-trunc.** uses the DTR linearization that truncates cell for input, **all-table-trunc.** discard excess cells to input all contents using 512 tokens. **partial** selects random rows to fits 100 words.

In the second method, we adopt a similar row-wise approach, but (1) limit the number of tokens in header and content cells to 12 and 8 [4], given that the previous dynamic truncation strategy is risky of truncating cells too aggressively for long tables, and that header information is often more crucial; (2) only include the first few rows instead of all rows that fit into the 512-token limit. We denote this by **all-table-trunc.** in Tab. 4.

Meanwhile, regarding that DPR is accustomed to 100-word text passages during training, our third experiment fits table linearization into the 100-word input limit by selecting the first few rows [5], which is denoted as **partial** in Tab. 4. We aim to study if a moderate compromise of reducing content to meet the optimal sequence length helps.

Results of three experimental settings are shown in Tab. 4. First, as shown by the **no-content** results, using the title-header combination can solve a decent amount of cases. Compared to the solvable cases with additional table content, this content-free setting can solve around 65% to 80% of them, if examined among the top 1-5 metrics. Second, comparing **all-cell-trunc.** and **all-row-trunc.** that write all table content using 512-tokens, our second method that removes end rows is better than the DTR strategy of cell truncation. Trading off cell lengths to fit more cells deteriorates the quality of generated representations. Third, the **partial** method performs better than the above two using all table content. Trimming tables to conform to the training format (100-word), even by discarding some of the content rows, can be more helpful than naively enlarging the content volume.

### 5.2 Perturbing the Structured Table Layout

Results in § 4.2 suggests that a retriever model without explicit structural embeddings can perform similarly well as one that does. Although the DPR architecture has no explicit features to encode structures such as row and column positions, it can potentially combine the BERT sequential position indices with cell/row separators to configure the cell mapping as in a two-dimensional table layout. In this way, DPR can implicitly capture some structures using a generic sequence-oriented encoder.

Therefore, we aim to investigate if the DPR representation on table structure is sufficient, or that NQ-table retrieval actually relies more on the table structural information. To prove which is the case, we propose to perturb the structured table cell mapping by shuffling the order of table cells.

**Shuffle In Row** A table row often relates to the same entity and contains multiple cells to describe the various properties of that entity. The content

[4] We experiment with cell limit 6/8/10 and header limit 8/12/16 and find this to be the best configuration.

[5] Using the first few or a random selection of rows performs very similarly in our preliminary experiments

of each cell maps with the header cell in the corresponding columns. In this setting, we randomly select one row in the table and shuffle all of its cells, such that the resulting cell order in the selected row does not match it in the header row. In other words, for each cell in the selected row, the type declared in the header cell of that column does not match with it. We denote it as **DPR-row** in Tab. 5.

**Shuffle In Column** Table cells within the same column are often different values given the type description of the column header cell. We also experiment with the shuffle-in-column setting, where in a randomly selected column all cells are re-ordered. This setting is denoted as **DPR-col** in Tab. 5.

| Model | Retrieval Accuracy | | | | | |
|---|---|---|---|---|---|---|
| | @1 | @3 | @5 | @10 | @20 | @50 |
| DPR | 61.38 | **80.54** | **85.51** | **91.41** | **94.62** | 96.69 |
| DPR-row | 60.97 | 79.30 | 84.99 | 91.20 | 94.20 | 97.00 |
| DPR-col | 60.86 | 79.30 | 84.99 | 91.10 | 94.31 | **97.31** |

Table 5: Zero-shot table retrieval accuracy measured in top-k metrics on NQ-table test. DPR linearizes tables in the correct order. For each table, *DPR-row* perturbs the order of cells within one row, *DPR-col* perturbs the order of cells within one column.

From the results in Tab. 5, perturbing orders, either row-wise or column-wise, only lead to minor changes in the retrieval accuracy. For the questions in the NQ-table dataset, retrieving the correct table does not heavily require on its structured layout.

## 6 Related Work

**Open Domain Question Answering (ODQA)** ODQA systems often use a retriever-reader pipeline, where the retriever selects relevant contexts and the reader further selects answer spans from them. Because candidate contexts normally approach millions in size, good retrieval accuracy is critical for an effective end-to-end QA system (Karpukhin et al., 2020). Most text retrieval systems are built on the Dense Passage Retriever (DPR) (Karpukhin et al., 2020). Beyond texts, one of the most common sources for answering open-domain questions is structured tables. Herzig et al. (2021) recently identified a subset of Natural Questions (NQ) dataset (Kwiatkowski et al., 2019) that is answerable by Wikipedia tables. Oguz et al. (2021); Ma et al. (2021) find Wikipedia tables also helpful to answer NQ questions. This paper aims to optimize the usage of tables for QA and focuses on the task of table retrieval.

**Table Understanding** To encode the relational structure of web tables, CNNs (Chen et al., 2019a), RNNs (Gol et al., 2019), LSTMs (Fetahu et al., 2019), and their combinations (Chen et al., 2019b) are explored. In addition, Graph Neural Network (GNN) is used, especially for tables with complex structures (Koci et al., 2018; Zayats et al., 2021; Vu et al., 2021; Bhagavatula et al., 2015). Upon the advances in pre-trained Transformers, recent table encoders adapt the BERT model with table-specific modules concerning structure (Herzig et al., 2020; Yin et al., 2020; Wang et al., 2021b), numeracy (Wang et al., 2021b; Herzig et al., 2020), and other features. These methods are intentionally built for tables, but their necessity in certain tasks remains unknown. Our work exploits a generic model to show that some content-emphasized tasks like retrieval do not require such specific designs.

**Table Retrieval** Earlier work focus on web table search in response to a keyword query (Cafarella et al., 2008, 2009; Balakrishnan et al., 2015; Pimplikar and Sarawagi, 2012) or a seed table (Sarmad et al., 2012). Many of them use the 60 keywords and relevant web tables collected by Zhang and Balog (2018). Tables are modeled by aggregating multiple fields (Zhang et al., 2019), contexts (Trabelsi et al., 2019), and synthesized schema labels (Chen et al., 2020b). More recently, Chen et al. (2020c); Wang et al. (2021a) use structure-augmented BERT for retrieval. These works largely treat the retrieval task on its own account and target similarity under the traditional Information Retrieval (IR). We focus on the extended scope of retrieval to also emphasize model inference abilities for answering questions in the open domain.

## 7 Conclusion

Given the importance of finding relevant tables to answer questions in the NQ-table dataset, we study the task of table retrieval and identify question-table matching patterns that emphasize content rather than table structure. Our experiment using a text-generic Dense Passage Retriever (DPR) proves its comparable performance to the state-of-the-art Dense Table Retriever (DTR) on table retrieval. While both adding table-specific modules and perturbing table cell ordering in DPR bring negligible change, our work reveals the unnecessity of table-specific designs for table retrieval, as well as the great potential of generic text retrievers in their direct applicability on structured tables.

## References


Sreeram Balakrishnan, Alon Halevy, Boulos Harb, Hongrae Lee, Jayant Madhavan, Afshin Rostamizadeh, Warren Shen, Kenneth Wilder, Fei Wu, and Cong Yu. 2015. Applying webtables in practice.

Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. 2015. Tabel: Entity linking in web tables. In *International Semantic Web Conference*, pages 425–441. Springer.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Michael J Cafarella, Alon Halevy, and Nodira Khoussainova. 2009. Data integration for the relational web. *Proceedings of the VLDB Endowment*, 2(1):1090–1101.

Michael J Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. 2008. Webtables: exploring the power of tables on the web. *Proceedings of the VLDB Endowment*, 1(1):538–549.

Jiaoyan Chen, Ernesto Jiménez-Ruiz, Ian Horrocks, and Charles Sutton. 2019a. Colnet: Embedding the semantics of web tables for column type prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 29–36.

Jiaoyan Chen, Ernesto Jiménez-Ruiz, Ian Horrocks, and Charles Sutton. 2019b. Learning semantic annotations for tabular data. *arXiv preprint arXiv:1906.00781*.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Wang. 2020a. Hybridqa: A dataset of multi-hop question answering over tabular and textual data. *arXiv preprint arXiv:2004.07347*.

Zhiyu Chen, Haiyan Jia, Jeff Heflin, and Brian D Davison. 2020b. Leveraging schema labels to enhance dataset search. *Advances in Information Retrieval*, 12035:267.

Zhiyu Chen, Mohamed Trabelsi, Jeff Heflin, Yinan Xu, and Brian D Davison. 2020c. Table search using a deep contextualized language model. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 589–598.

Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2020. Turl: Table understanding through representation learning. *arXiv preprint arXiv:2006.14806*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Besnik Fetahu, Avishek Anand, and Maria Koutraki. 2019. Tablenet: An approach for determining fine-grained relations for wikipedia tables. In *The World Wide Web Conference*, pages 2736–2742.

Majid Ghasemi Gol, Jay Pujara, and Pedro Szekely. 2019. Tabular cell classification using pre-trained cell embeddings. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 230–239. IEEE.

Jonathan Herzig, Thomas Müller, Syrine Krichene, and Julian Martin Eisenschlos. 2021. Open domain question answering over tables via dense retrieval. *arXiv preprint arXiv:2103.12011*.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Mueller, Francesco Piccinno, and Julian Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333.

Sujay Kumar Jauhar, Peter Turney, and Eduard Hovy. 2016. Tables as semi-structured knowledge for question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 474–483.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781.

Elvis Koci, Maik Thiele, Wolfgang Lehner, and Oscar Romero. 2018. Table recognition in spreadsheets via a graph representation. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pages 139–144. IEEE.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096.

Qian Liu, Bei Chen, Jiaqi Guo, Zeqi Lin, and Jianguang Lou. 2021. Tapex: Table pre-training via learning a neural sql executor. *arXiv preprint arXiv:2107.07653*.

Kaixin Ma, Hao Cheng, Xiaodong Liu, Eric Nyberg, and Jianfeng Gao. 2021. Open domain question answering over virtual documents: A unified approach for data and text. *arXiv preprint arXiv:2110.08417*.

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.

Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Scott Yih. 2021. Unik-qa: Unified representations of structured and unstructured knowledge for open-domain question answering. *arXiv preprint arXiv:2012.14610*, 54:57–60.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480.

Rakesh Pimplikar and Sunita Sarawagi. 2012. Answering table queries on the web using column keywords. *Proceedings of the VLDB Endowment*, 5(10):908–919.

Anish Das Sarmad, Lujun Fang, Nitin Guptad, Alon Halevyd, Hongrae Leed, Fei Wud, Reynold Xin, and Cong Yud. 2012. Finding related tables.

Mohamed Trabelsi, Brian D Davison, and Jeff Heflin. 2019. Improved table retrieval using multiple context embeddings for attributes. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 1238–1244. IEEE.

Binh Vu, Craig A Knoblock, Pedro Szekely, Minh Pham, and Jay Pujara. 2021. A graph-based approach for inferring semantic descriptions of wikipedia tables. In *International Semantic Web Conference*, pages 304–320. Springer.

Fei Wang, Kexuan Sun, Muhao Chen, Jay Pujara, and Pedro Szekely. 2021a. Retrieving complex tables with multi-granular graph representation learning. *arXiv preprint arXiv:2105.01736*.

Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. 2021b. Tuta: Tree-based transformers for generally structured table pre-training. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1780–1790.

Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. 2020. Break it down: A question understanding benchmark. *Transactions of the Association for Computational Linguistics*, 8:183–198.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. Tabert: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8413–8426.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Caiming Xiong, et al. 2020. Grappa: Grammar-augmented pre-training for table semantic parsing. In *International Conference on Learning Representations*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921.

Vicky Zayats, Kristina Toutanova, and Mari Ostendorf. 2021. Representations for question answering from documents with tables and text. *arXiv preprint arXiv:2101.10573*.

Li Zhang, Shuo Zhang, and Krisztian Balog. 2019. Table2vec: Neural word and entity embeddings for table population and retrieval. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1029–1032.

Shuo Zhang and Krisztian Balog. 2018. Ad hoc table retrieval using semantic similarity. In *Proceedings of the 2018 world wide web conference*, pages 1553–1562.

Shuo Zhang and Krisztian Balog. 2020. Web table extraction, retrieval, and augmentation: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(2):1–35.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

## A Model Size

Tab. 6 shows the detailed configurations of BERT-variants in different sizes. As can be seen from the hyper-parameter values, models of medium size have the smallest capacity, base is an intermediate configuration, and large size is the biggest.

| Size | Layers | Attention Heads | Hidden Size |
|---|---|---|---|
| medium | 8 | 8 | 512 |
| base | 12 | 8 | 768 |
| large | 24 | 16 | 1024 |

Table 6: Hyper-parameters for BERT models of varied sizes. Models of different sizes vary in the number of Transformer layers, the number of heads in the self-attention module, and the dimension of hidden states.

## B Experiment Details

For the DPR experiments, we use the latest published checkpoint (in base size) [6] where the hard-negative text passages are mined using the DPR checkpoint saved in the previous round. Except for our experiments on table linearization strategies, all data processing details follow the original DPR manipulation.

To substitute the original DPR context encoder with the one with augmented embeddings (TAPAS), we import the implemented version in the Huggingface [7] repository. Following the default setting of DPR, all fine-tuning use batch size $= 16$ and learning rate $= 2\mathrm{e}{-5}$. Experiments run on a single GPU on the Tesla K80 server.

To reproduce the DTR performance, we use the published checkpoints (in medium and large sizes) [8] and run the retrieval inference. Our results are comparably close to the number reported in the paper and the GitHub repository, although they do not exactly match.

[6] https://github.com/facebookresearch/DPR

[7] https://github.com/huggingface/transformers

[8] https://github.com/google-research/tapas/blob/master/DENSE_TABLE_RETRIEVER.md