Table Retrieval May Not Necessitate Table-specific Model Design

Zhiruo Wang,   Zhengbao Jiang,   Eric Nyberg,   Graham Neubig

Language Technologies Institute, Carnegie Mellon University

{zhiruow,zhengbaj,ehn,gneubig}@cs.cmu.edu

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 has designed special models and training paradigms for table-related tasks such as table-based question answering and table retrieval. Though effective, they add complexity in both modeling and data acquisition compared to generic text solutions and obscure which elements are truly beneficial. In this work, we focus on the task of table retrieval, and ask: "is table-specific model design necessary for table retrieval, or can a simpler text-based model be effectively used to achieve a similar result?" First, we perform an analysis on a table-based portion of the Natural Questions dataset (NQ-table), and find that structure plays a negligible role in more than 70% of the cases. Based on this, we experiment with a general Dense Passage Retriever (DPR) based on text and a specialized Dense Table Retriever (DTR) that uses table-specific model designs. We find that DPR performs well without any table-specific design and training, and even achieves superior results compared to DTR when fine-tuned on properly linearized tables. We then experiment with three modules to explicitly encode table structures, namely auxiliary row/column embeddings, hard attention masks, and soft relation-based attention biases. However, none of these yielded significant improvements, suggesting that table-specific model design may not be necessary for table retrieval.[1]

1 Introduction

Tables are a valuable form of data that organize 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 organized in a more accessible manner than in unstructured texts, or even not

---

[1]The code and data are available at https://github.com/zorazrw/nqt-retrieval

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]
Zack Greinke	SP	Los Angeles Dodgers	$25,000,000	[19]
Felix Hernandez	SP	Seattle Mariners	$24,907,142	[20]

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

available in text at all (Chen et al., 2020a). 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 and 74.4% by texts.

Because tables are intuitively different from unstructured text, most previous works consider text-based methods to be functionally incapable of processing tables effectively and create special-purpose models with table-specific architectures and training methods, adding auxiliary structure-indicative parameters (Herzig et al., 2020; Wang et al., 2021b; Deng et al., 2020; Yang et al., 2022), enforcing structure-aware attention (Yin et al., 2020; Wang et al., 2021b; Zayats et al., 2021), and table-oriented pre-training 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 encod-

ing, and must be intentionally built for and trained on tabular 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, we hypothesize that content matching is paramount, and little, if any, structural understanding may be required. For example, given a question "Who is the highest paid baseball player in the major leagues?" in Figure 1, a correct table can be retrieved by simply identifying the phrase "highest-paid", "major league", and "baseball player" in the table title, and matching the semantic type of "Who" to the "Name" header. Hence, any benefit demonstrated by table-based models may well come from good training data while table-specific model design has a limited influence.

In this paper, we specifically ask: "Does table retrieval require table-specific model design, or can properly trained generic text retrievers be exploited to achieve similar performance with less added complexity?" Our work centers around the table-based open-domain QA dataset, NQ-table (Herzig et al., 2021), a subset of the Natural Questions (NQ) dataset (Kwiatkowski et al., 2019) where each question can be answered by part(s) of a Wikipedia table. We start with manual analysis of 100 random samples from NQ-table and observe that consideration of table structure seems largely unnecessary in over 70% of the cases, while the remaining 30% of cases only require simple structure understanding such as row/column alignment without structure-dependent complex reasoning chains (§ 2). With this observation, we experiment with two strong retrieval models: a general-purpose text-based retriever (DPR; Karpukhin et al. (2020)) and a special-purpose table-based retriever (DTR; Herzig et al. (2021)). We find that DPR, without any table-specific model design or training, achieves similar accuracy as the state-of-the-art table retriever DTR, and further fine-tuning on NQ-table yields significantly superior performance, casting doubt on the necessity of table-specific model design in table retrieval (§ 3). Using DPR as the base model, we then thoroughly examine the effectiveness of both encoding structure implicitly with structure-preserving table linearization (§ 4) and encoding structure explicitly with table-specific model design, such as auxiliary embeddings and specialized attention mechanisms (§ 5).

We find that models can already achieve a degree of structure awareness using properly linearized tables as inputs, and additionally adding explicit structure encoding model designs does not yield a further improvement. In sum, the results reveal that a strong text-based model is competitive for table retrieval, and table-specific model designs may have limited additional benefit. This indicates the potential to directly apply future improved text retrieval systems for table retrieval, a task where they were previously considered less applicable.

2 NQ-table Analysis: How Much Structure Does Table Retrieval Require?

The NQ-table dataset (Herzig et al., 2021) is a subset of the Natural Questions (NQ) dataset (Kwiatkowski et al., 2019) which contains questions from real users that can be answered by Wikipedia articles. Previous works on text-based QA extract the text portion from source Wikipedia articles that can answer around 71k questions, while NQ-table extract tables that contain answers for 12k questions. Unless otherwise specified, we use NQ-text to denote the commonly referred NQ dataset that can be answered by texts.

To better understand to what extent (if any) is structure understanding required by table retrieval, we perform a manual analysis on the NQ-table dataset. Specifically, we randomly sample 100 questions and their relevant tables then categorize their matching patterns.

Keyword Matching Without Structural Concern Aligning with the insight that retrieval often emphasizes content matching rather than complex reasoning (Rogers et al., 2021), we find that 71 out of the 100 samples only require simple keyword matching, where 18 questions fully match with table titles (Figure 2 (a)) and the other 53 questions further match with table headers (Figure 2 (b)).

Retrieval that Requires Row/Column Alignment For the other 29 samples, understanding table structure is helpful but only simple row/column alignment is needed. 21 of them require locating content cells in a specific column and combining the information from headers. For example in Figure 2(c), under the general header "Population", one should locate the "Total" field by their structural relation to confirm that the 'total number' measure of 'population' exists. In addition, 7 of the samples are some-

Question:
What is the largest man made lake in the us?

List of largest lakes of the United States by area					
Rank ⬍	Name ⬍	U.S. states/Canadian provinces/Mexican states ⬍	Area	⬍	Type ⬍
1	Lake Superior	Michigan–Minnesota–Wisconsin–Ontario	31,700 sq mi	82,103 km²	natural
2	Lake Huron	Michigan–Ontario	23,000 sq mi	59,570 km²	natural
3	Lake Michigan	Illinois–Indiana–Michigan–Wisconsin	22,300 sq mi	57,757 km²	natural
		......			
7	Lake of the Woods	Manitoba–Minnesota–Ontario	1,679 sq mi	4,349 km²	natural
8	Iliamna Lake	Alaska	1,014 sq mi	2,626 km²	natural
9	Lake Oahe	North Dakota–South Dakota	685 sq mi	1,774 km²	man-made
10	Lake Okeechobee	Florida	662 sq mi	1,715 km²	natural

(a)

Question:
What is the genus of a bald eagle?

Bald eagle	
Kingdom:	Animalia
Phylum:	Chordata
	......
Genus:	Haliaeetus
Species:	H. leucocephalus

(b)

Question:
What is the population of Florida?

Keystone Heights, Florida	
......	
Country	United States
State	Florida
County	Clay
Population (2020)	
· Total	1,446
· Density	1,345.12/sq mi (519.52/km²)

(c)

Figure 2: Table (a) matches the question by its title, (b) matches topic in title and answer type in header, and in (c) knowing the column alignment helps.

what ambiguous and may require external knowledge or question clarification (Min et al., 2020).

In summary, our analysis reveals that understanding table structure is not necessary in the majority of cases, and even for cases where structural information is useful, they merely require aligning the rows/columns instead of building complex chains of reasoning.

3 Text Retrieval vs Table Retrieval

Given the previous analysis, we hypothesize that general-purpose text-based retrievers without table-specific designs 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). Properly trained text-based retrievers might even outperform table-based retrievers because the strong content matching ability learned on text retrieval datasets can transfer to the table retrieval task.

To validate these assumptions, we examine two representative retrieval systems: the text-based Dense Passage Retriever (DPR) and the table-based Dense Table Retriever (DTR). We first briefly introduce their input formats and model architectures (§ 3.1, § 3.2), then conduct experiments in both zero-shot and fine-tuning settings and compare their table retrieval performance (§ 3.3).

3.1 Text Retriever: DPR

We choose DPR (Karpukhin et al., 2020) as a representative text 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 Table 1. The question encoder BERT_q encodes each question q and outputs its dense representation using the representation of [CLS] token, denoted as $h_q = \text{BERT}_q(q)[\text{CLS}]$. The context encoder works similarly. To enable tables for sequential context inputs, we linearize each table into a token sequence T, which is then fed into the context encoder BERT_c to obtain its dense representation $h_T = \text{BERT}_c(T)[\text{CLS}]$. The similarity score between a question q and a table T is computed as the dot product of two vectors $sim(q, T) = h_q \cdot h_T$.

DPR has been trained only on sequential text contexts. 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 annotated correct contexts and mined hard negative contexts.

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 delimiter). Within each header or content row, we concatenate adjacent cell strings using a vertical bar '|' (cell delimiter). A template table linearization reads as [title].[header].[content₁]. ⋯ .[contentₙ]. Although the BERT encoder has the capacity for a maximum of 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 selecting the first few rows that fit into the 100-word window.

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.

Model Architecture DTR largely follows the bi-encoder structure of DPR, but differs from it in the embedding layer. As shown in Table 1, DTR utilizes the existing embeddings in alternative ways

and introduces new types of embeddings specifically designed to encode tables.

Both models use the BERT vocabulary index for token embedding. For the segment index, DPR assigns all tokens in a sequence to index 0, while DTR distinguishes the title from table content by assigning 0 and 1, respectively. For positions, DPR inherits from BERT the sequence-wise order index $[0, 1, 2, ..., \text{sequence length}-1]$; DTR adopts a cell-wise reset strategy that records the index of a token within its located cell $[0, 1, ..., \text{cell length} - 1]$.

Most importantly, DTR introduces row and column embeddings to encode the structural position of each token in the cell that it appears. This explicit join of three positional embeddings is potentially more powerful than the BERT-style flat index. Besides, concerning the high frequency of numerical values in tables, DTR adds a ranking index for each token if it is part of a number.

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.

Training Process DTR also has a more complex training process than DPR. As summarized in Figure 3, DTR has a three-stage training using tables.
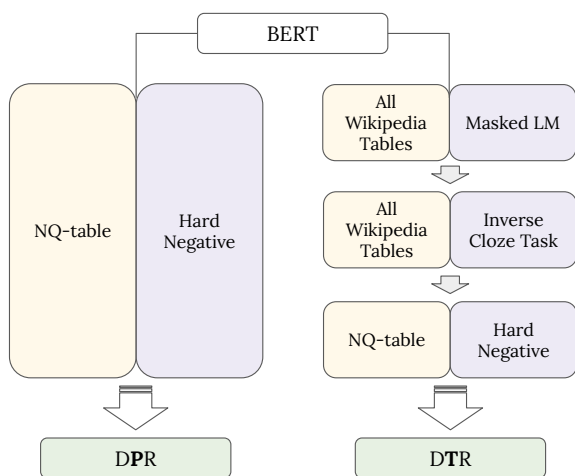

Figure 3: Comparison of DPR and DTR training.

First, model parameters, except for those extra table-specific embeddings, are initialized with BERT weights. The model is then pre-trained on all Wikipedia tables using the Masked LM (MLM) (Devlin et al., 2018) task, yielding the TAPAS (Herzig et al., 2020) model. Second, to leverage TAPAS to the retrieval task, 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 is trained 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 Text Retrieval Benefits Table Retrieval

To evaluate the benefit on table retrieval from training on in-domain text retrieval datasets, we compare the performance of DPR and BERT (Devlin et al., 2018) after fine-tuning on NQ-table.

As shown in Table 3, *BERT-table* significantly underperforms *DPR-table*, indicating that training on in-domain text retrieval datasets benefits the table retrieval task. We conjecture that the large gap is essentially because (1) NQ-text and NQ-table questions share similar characteristics hence are agnostic to the format of answer source (Wolfson et al., 2020), and (2) NQ-text has a larger size than NQ-table (71k versus 12k).

3.4 DPR vs DTR

To verify if table-specific model designs in DTR are necessary, we start with comparing the original DPR with DTR to evaluate their off-the-shelf performance, then proceed to fine-tune DPR on NQ-table to examine the how much improvement can be brought by training data. We evaluate both models on NQ-table test set and measure the retrieval accuracy by computing the portion of questions where the top-k retrieved tables contain the answer.

For DPR experiments, we use the latest published checkpoint[2] where the hard-negative text passages are mined using the DPR checkpoint saved in the previous round. To reproduce the DTR performance, we use the published checkpoints[3] and run the retrieval inference.

To curate training samples for questions in the NQ-table training set, we take the same positive table used in DTR training. For negative contexts,

---

[2]https://github.com/facebookresearch/DPR
[3]https://github.com/google-research/tapas/blob/master/DENSE_TABLE_RETRIEVER.md

we use the original DPR checkpoint to retrieve the top-100 table candidates for each question, from which we take the highest-ranked tables without answers as the hard negatives. We train with a batch size of 16 and a learning rate of $2\mathrm{e}-5$. Experiments are finished on four NVIDIA Tesla V100 GPUs.

Note that the published DPR and DTR checkpoints are not strictly comparable, since the size of DPR base falls between the DTR medium and DTR large with respect to the number of parameters. We report the performance of DTR in both medium and large size to approximate the lower and upper bounds for the DTR base model.

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

Table 2: 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.

Table 2 shows the 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.

As reported in Table 3, DPR is able to achieve a zero-shot retrieval accuracy (*DPR*) on NQ-table that is fairly close to the state-of-the-art DTR model, even without any table-specific model design and training. Further, simply fine-tuning DPR on NQ-table (*DPR-table*) using the same annotated positive and mined hard-negative tables as DTR increases the performance by a large margin, achieving superior performance than DTR, especially at top ranking positions (i.e., small k).

Model	Retrieval Accuracy				
	@1	@5	@10	@20	@50
DTR (medium)	62.32	82.51	86.75	91.51	94.26
DTR (large)	63.98	84.27	**89.65**	**93.48**	**95.65**
BERT-table	60.97	79.81	85.51	88.20	91.62
DPR	57.04	80.54	86.13	89.54	92.34
DPR-table	**67.91**	**84.89**	88.72	90.58	92.86

Table 3: Top-k table retrieval accuracy on NQ-table test set. *DPR* is the original model checkpoint. *DPR-table* and *BERT-table* are DPR and BERT fine-tuned on NQ-table respectively.

These observations question the necessity of both table-specific model designs listed in Table 1 and table-specific pre-training listed in Figure 3. Given the task analysis in § 2 that table retrieval only requires simple structure understanding, we hypothesize that DPR, trained with table inputs linearized from top-to-bottom and left-to-right, is functionally capable of implicitly encoding simple table structure such as row/column alignment, and the benefit of extra table-specific model designs is minimal. To thoroughly and rigorously verify our hypothesis, we first examine the effect of different ordering in table linearization in § 4, then experiment with three widely-used structure injection model designs by adding them on DPR in § 5.

4 Implicit Structure Encoding from Linearized Tables

The simplest way to encode table structure is to linearize the table following the top-to-bottom left-to-right order and insert delimiters between cells and rows, from which the sequence-oriented transformer models should also be able to recover the two-dimensional table structure.

We hypothesize that this type of implicit structure encoding is sufficient for table retrieval, which only requires simple structure understanding. To verify this, we manipulate linearized tables by randomly shuffling their rows/columns (§ 4.1) or removing the delimiters (§ 4.2), and examine how these perturbation affect the final performance.

4.1 Shuffling Rows and Columns

Our first experiment focuses on the order of table linearization: if DPR relies on a proper linearization to capture table structure, randomly shuffling the table contents should corrupt the structure information and hurt the representation quality, leading to lower retrieval accuracy.

To verify this, we shuffle table cells within each *row*, each *column*, or *both*. Cells in the same row often describe the same entity from multiple properties according to their column headers, therefore shuffling the order of multiple cells in the same row corrupts their alignment with header cells. Meanwhile, cells in the same column are often of the same semantic type but are attributes to different entities in different rows, shuffling the order of cells in the same column breaks their alignment with entities. We also examine shuffling on both dimensions, which completely removes the order

information from the table linearizations.

Since models trained on properly linearized tables might be prone to the train-test discrepancy when tested on shuffled tables, we conjecture that the gap between testing on proper tables and shuffled tables cannot be fully attributed to the loss of order information. We therefore conduct a more rigorous experiment by fine-tuning DPR on shuffled tables in both dimensions (*DPR-table w/ shuffle*) and test it on both proper and shuffled tables.

Method		Retrieval Accuracy				
Model	Shuffle	@1	@5	@10	@20	@50
DPR	-	57.04	80.54	86.13	89.54	92.34
	row	55.18	79.19	85.82	89.75	92.44
	column	57.04	80.85	86.65	89.34	92.55
	both	57.97	79.61	84.89	89.44	92.55
DPR-table	-	67.91	84.89	88.72	90.58	92.86
	row	55.18	76.09	80.64	85.40	89.23
	column	58.39	77.74	82.92	86.44	89.86
	both	54.76	75.16	80.64	84.87	88.82
DPR-table w/ shuffle	-	62.94	80.12	84.99	88.92	91.30
	row	62.11	80.95	85.30	88.82	91.72
	column	64.91	82.30	86.75	89.54	92.55
	both	63.35	81.06	85.20	89.34	91.93

Table 4: Top-k table retrieval accuracy on shuffled NQ tables, using the original *DPR*, the fine-tuned (*DPR-table*), and the fine-tuned on shuffled tables (*DPR-table w/ shuffle*).

As shown in Table 4, on the original DPR model, all table shuffling strategies result in minor variations in retrieval accuracy, which is intuitive because DPR has never been trained on linearized tables and it is not sensitive to cell orders.

The performance of fine-tuned DPR (*DPR-table*) drops significantly when tested on shuffled tables, similar to the previous finding that T5 model is also sensitive to the ordering of structured knowledge (Xie et al., 2022). Besides the potential discrepancy in table layout between training and test inputs, this may indicate that DPR, although without explicit structure encoding modules, also learns to implicitly capture structures by training on linearized table inputs.

To ablate out the influence of train-test discrepancy, we also fine-tune DPR on shuffled positive and negative tables. As expected, *DPR-table w/ shuffle* does not suffer from train-test discrepancy. While DPR fine-tuned on shuffled tables still outperforms the original DPR (57.04→62.94@1), the improvement is not as significant as the im-

provement obtained by fine-tuning on proper tables (57.04→67.91@1), indicating that DPR is able to utilize structure-preserving table linearizations to encode structures during training.

Comparing different shuffling dimensions, we notice that in-*row* shuffling hurts the performance more than in-*column* shuffling, indicating that preserving semantic type alignment within each column is more important than preserving entity alignment within each row for table retrieval.

4.2 Removing Delimiters Between Rows/Cells

In this section, we study the impact of delimiters in helping models to encode table structures. If delimiters are not included, it is theoretically impossible to recover the table structure even from properly linearized tables, because the boundaries between different cells and rows are unknown. To verify if delimiters can serve as effective indicators of table structure, we study the usefulness of both inserting delimiter ('|') between cells and inserting delimiter ('.') between rows.

Similarly to the previous experiment, we evaluate (1) the original DPR model (*DPR*), (2) the DPR fine-tuned on tables with delimiters (*DPR-table*), and (3) the one fine-tuned on linearized tables without delimiters (*DPR-table w/o delimiter*).

Method		Retrieval Accuracy				
Model	Delimiter	@1	@5	@10	@20	@50
DPR	all	57.04	80.54	86.13	89.54	92.34
	cell	56.00	79.40	85.30	89.54	92.34
	row	54.24	77.54	82.92	87.68	92.13
	none	55.49	79.09	84.78	89.44	92.03
DPR-table	all	67.91	84.89	88.72	90.58	92.86
	cell	55.80	75.26	81.16	85.20	89.23
	row	55.07	74.95	80.75	84.68	89.65
	none	56.63	76.19	81.26	86.13	89.75
DPR-table w/o delimiter	all	63.46	81.47	85.09	88.82	92.13
	cell	63.04	83.02	87.47	90.06	92.13
	row	63.35	80.54	85.20	89.34	92.03
	none	64.49	81.88	86.23	89.86	92.55

Table 5: NQ-table retrieval accuracy with linearized table w/ and w/o cell and row delimiters. *cell* linearizes table by only inserting delimiters between cells, *row* only inserts delimiters between rows, and *none* inserts neither.

As shown in Table 5, for *DPR*, although the overall performance drop is small without delimiters, separating cells is more important than separating rows, which is intuitive because the number of cells is larger than the number of rows. On *DPR-table*

that learns from properly delimited tables, the influence is more significant, and the extent of dropping is similar to that of table structure shuffling in Table 4. Also similar to the previous findings, training on non-delimited tables (*DPR-table w/o delimiter*) improves over the original DPR, but the improvement is not as significant as the improvement obtained by fine-tuning on delimited tables, suggesting that cell and row delimiters help models encode table structure.

5 Explicit Structure Encoding with Table-specific Model Design

From the previous section, we conclude that DPR can already encode simple table structures based on structure-preserving linearized tables with correct cell order and delimiters. The next question is "can explicit table-specific model designs encode more complex structure that is useful beyond the capacity of implicit encoding?"

In this section, we examine three widely used table-specific modules to explicitly encode table structure information by adding these modules on top of the DPR architecture. As summarized in Table 6 and illustrated in Figure 4, we categorize existing methods for table-specific structure encoding into three representative types: (1) auxiliary table-specific embeddings, (2) restricted hard attention mask to enforce structure-aware attention, and (3) soft attention bias based on the structural relations of cell pairs. For each component, we add it onto the DPR architecture and fine-tune under the same setting as for *DPR-table*.

Method	Papers
auxiliary embeddings	TAPAS (Herzig et al., 2020) MATE (Eisenschlos et al., 2021) TUTA (Wang et al., 2021b) TABBIE (Iida et al., 2021)
hard attention mask	TURL (Deng et al., 2020) SAT (Zhang et al., 2020) ETC (Ainslie et al., 2020) DoT (Krichene et al., 2021) MATE (Eisenschlos et al., 2021) TUTA (Wang et al., 2021b)
soft attention bias	RAT-SQL (Wang et al., 2020) TableFormer (Yang et al., 2022)

Table 6: Structure encoding methods used in previous works for table-related tasks.

5.1 Auxiliary Row and Columns Embeddings

We first examine if adding table-specific embedding parameters would bring additional improvement. Specifically, we add row and column embeddings into the DPR to encode the row and column indices of tokens, which is denoted as *DPR-table w/ emb*. Both row and column indices are 1-indexed, and 0 is used for tokens that are not part of the table (e.g., title). We initialize row/column embeddings with zero to allow smooth continual learning.

5.2 Hard Attention Mask

Another approach is to enforce structure-aware attention using hard attention mask that only allows attention between elements within their mutual structural proximity, with the assumption that elements are only semantically relevant to elements in their structural proximity. Specifically, Krichene et al. (2021); Eisenschlos et al. (2021); Deng et al. (2020) sparsify the attention mask such that each token is only visible to other tokens that are either within the same row or the same column. We apply this masking strategy when fine-tuning DPR and denote this setting as *DPR-table w/ mask*.

5.3 Soft Relation-based Attention Bias

The third method is to bias the attention weight between two tokens based on their structural relation, which is a more fine-grained way to enforce structure-aware attention than hard mask. Specifically, different bias scalars are added to the attention scores based on the relation between two cells. Wang et al. (2020) categorize relations by columns, while Yang et al. (2022) defines 13 relations based on which component the token belongs to: sentence, header, and cell. A more concrete example is illustrated in Figure 4. Relational bias is invariant to the numerical indices of rows and columns, which is more robust to answer-invariant structure perturbation. We follow Yang et al. (2022) to add soft attention bias on DPR with 13 relations.

5.4 Results and Analysis

As shown in Table 7, methods that explicitly encode table structures, either with additional row/column embeddings (*w/ emb*), hard attention mask (*w/ mask*), or soft relation-based attention bias (*w/ bias*), do not bring improvements over the *DPR-table* baseline, indicating that given the capacity of DPR in implicitly encoding structure
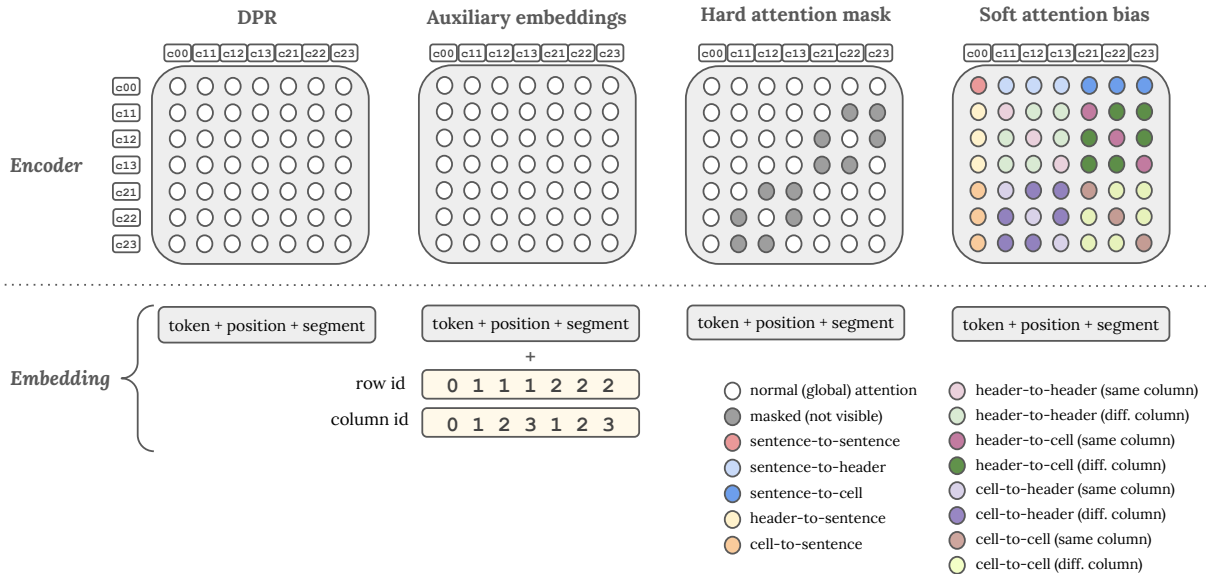
Figure 4: Illustration of three explicit structure encoding methods.

from linearized tables, the benefit of using special-purpose structure encoding modules is minimal.

Model	Retrieval Accuracy				
	@1	@5	@10	@20	@50
DPR-table	**67.91**	**84.89**	**88.72**	**90.58**	92.86
w/ emb	65.73	81.99	86.02	89.23	92.86
w/ mask	62.11	81.88	86.96	89.86	**93.06**
w/ bias	65.42	82.23	86.75	89.54	92.13

Table 7: Top-k table retrieval accuracy on NQ-table test. *DPR-table* is fine-tuned on the NQ-table without any table-specific modules, while the other three methods add auxiliary row/column embeddings (*w/ emb*), hard attention mask (*w/ mask*), and soft relation-based attention bias respectively (*w/ bias*).

6 Related Work

Open-domain Question Answering Open-domain QA systems often use a retriever-reader pipeline, where the retriever retrieves relevant contexts and the reader extracts or generates answer from them. Because the candidate context corpus is usually large with millions of documents, good retrieval accuracy is crucial for open-domain QA systems (Karpukhin et al., 2020). Beyond texts, another common sources for answering open-domain questions is 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) found that incorporating structured knowledge is beneficial for open-domain QA tasks.

Ma et al. (2021) showed that verbalizing structured knowledge into fluent text bring further gains over raw format for open-domain QA. Different from prior work, our paper analyzes different strategies for encoding tables with a focus on the task of table retrieval.

Table Understanding To encode the relational structure of 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 networks (GNN) are used, especially for tables with complex structures (Koci et al., 2018; Zayats et al., 2021; Vu et al., 2021; Bhagavatula et al., 2015). With the recent advances in pre-trained language models, table encoders adapt pre-trained language models with additional table-specific modules encoding structure (Herzig et al., 2020; Yin et al., 2020; Wang et al., 2021b) and numeracy (Wang et al., 2021b; Herzig et al., 2020). These methods are intentionally built for tables, but their necessity in each task remains unknown. Our work exploits a generic model to show that content-emphasized tasks like retrieval do not require such specific designs.

Table Retrieval Earlier works focus on web table search in response to keyword queries (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 mul-

tiple 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).

7 Conclusion

Given the importance of finding relevant tables when answering questions in the NQ-table dataset, we study the task of table retrieval and find that table retrieval emphasizes content rather than table structure. Our experiments with the text-generic Dense Passage Retriever (DPR) and the state-of-the-art table-specific Dense Table Retriever (DTR) demonstrate that DPR can already encode simple structures based on linearized tables and table-specific designs such as auxiliary embeddings, hard attention mask, and soft attention bias are not necessary. Our findings suggest that future development on table retrieval can potentially be built upon successful text retrievers and table-specific model designs should be carefully examined to avoid unnecessary complexity.

Acknowledgements

References

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. 2020. Etc: Encoding long and structured inputs in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 268–284.

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.

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*.

Julian Eisenschlos, Maharshi Gor, Thomas Mueller, and William Cohen. 2021. Mate: Multi-view attention for table transformer efficiency. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7606–7619.

Besnik Fetahu, Avishek Anand, and Maria Koutraki. 2019. Tablenet: An approach for determining finegrained 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.

Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer. 2021. Tabbie: Pretrained representations of tabular data. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3446–3456.

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.

Syrine Krichene, Thomas Mueller, and Julian Eisenschlos. 2021. Dot: An efficient double transformer for nlp tasks with tables. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3273–3283.

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 with a unified knowledge interface. *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.

Anna Rogers, Matt Gardner, and Isabelle Augenstein. 2021. Qa dataset explosion: A taxonomy of nlp resources for question answering and reading comprehension. *arXiv preprint arXiv:2107.12708*.

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.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578.

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.

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I Wang, et al. 2022. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. *arXiv preprint arXiv:2201.05966*.

Jingfeng Yang, Aditya Gupta, Shyam Upadhyay, Luheng He, Rahul Goel, and Shachi Paul. 2022. Tableformer: Robust transformer modeling for table-text encoding. *arXiv preprint arXiv:2203.00274*.

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*.

Hongzhi Zhang, Yingyao Wang, Sirui Wang, Xuezhi Cao, Fuzheng Zhang, and Zhongyuan Wang. 2020. Table fact verification with structure-aware transformer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1624–1629.

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*.