OPEN QUESTION ANSWERING OVER TABLES AND TEXT

Anonymous authors
Paper under double-blind review

ABSTRACT

In open question answering (QA), the answer to a question is produced by retrieving and then analyzing evidences that might contain answers to the question. Most open QA systems have considered only retrieving information from unstructured text only. Here we consider for the first time open QA over *both* tabular and textual data, and present a new large-scale dataset *Open Table-and-Text Question Answering (OTT-QA)* to evaluate performance on this task.[1] Most questions in OTT-QA require multi-hop inference across tabular data and unstructured text, and the evidence required to answer a question can be distributed in different ways over these two types of input, making evidence retrieval challenging. We also describe a system that introduces two novel techniques to address the challenge of retrieving and aggregating evidence for OTT-QA. The first technique is to use "early fusion" to group multiple highly relevant tabular and textual units into a fused block, which provides more context for the retriever to search. The second technique is to use a cross-block reader to model the cross-dependency between multiple retrieved evidences with global-local sparse attention. Collectively, these two methods improve end-to-end question answering performance significantly over the strong baselines by a margin over 15% exact match.

1 INTRODUCTION

An open QA model typically retrieves passages from a fixed text collection with a *retriever*, and then analyzes retrieved passages to provide answers to a given question with a *reader*. Prior open QA systems focused only on retrieving and reading unstructured text. However, a significant amount of information is stored in other formats, such as semi-structured web tables. For example, tables are often used to hold large quantities of related facts, especially numeric facts, such as `'Career Statistics for Lebron James'`. This type of detailed information is found much less frequently in unstructured text. Tables also are commonly used for collections of homogeneous entities or recurring events, like `'List of Periodic Comets'` or `'List of Champions League Winners since 1966'`. Hence tabular information provides an excellent complement to textual data, especially in the open setting. In spite of these advantages, no previous work has exploited the millions of web tables to augment the open QA system.

In this paper, we describe the first study to jointly exploit tables and text for open-domain question answering. For this purpose, we construct a new dataset, Open Table-and-Text Question Answering (OTT-QA). OTT-QA is built on the HybridQA dataset (Chen et al., 2020), and like HybridQA, OTT-QA questions are multi-hop questions which require information from both tables and text to answer. However, unlike HybridQA, OTT-QA requires the system to *retrieve* relevant tables and text—in contrast, in HybridQA, the tables and textual passages required for each question are given. To produce OTT-QA's questions, we begin by re-annotating the questions from HybridQA to 'de-contextualize' them—i.e., we make questions suitable for the open-domain setting so that unique answers can be determined from the question alone, rather than from the question together with the provided text and tables. We then add new questions to remove potential biases. After these steps, OTT-QA contains $45K$ human-annotated questions that require retrieving and aggregating information over tables and text from Wikipedia. Examples from OTT-QA are depicted in Figure 1. Note the table and passages contain non-overlapping information, and both of them must be understood

[1]All of the data will be released publicly.

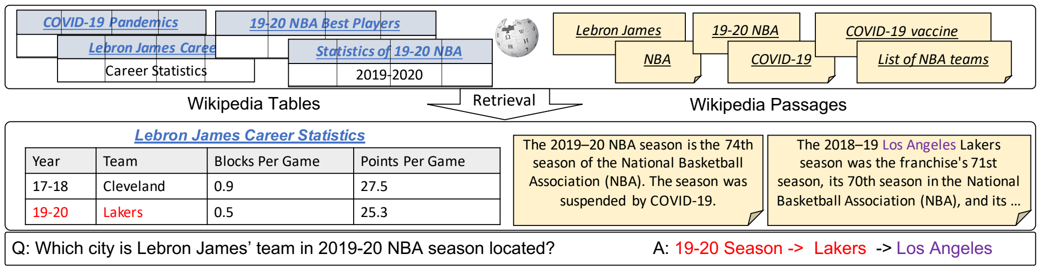

COVID-19 Pandemics
19-20 NBA Best Players
Lebron James Career
Career Statistics
Statistics of 19-20 NBA
2019-2020
Lebron James
19-20 NBA
COVID-19 vaccine
NBA
COVID-19
List of NBA teams
Wikipedia Tables
Retrieval
Wikipedia Passages
Lebron James Career Statistics
Year | Team | Blocks Per Game | Points Per Game
17-18 | Cleveland | 0.9 | 27.5
19-20 | Lakers | 0.5 | 25.3
The 2019–20 NBA season is the 74th season of the National Basketball Association (NBA). The season was suspended by COVID-19.
The 2018–19 Los Angeles Lakers season was the franchise's 71st season, its 70th season in the National Basketball Association (NBA), and its ...
Q: Which city is Lebron James' team in 2019-20 NBA season located?
A: 19-20 Season -> Lakers -> Los Angeles

Figure 1: The problem setting: A OTT-QA model needs to retrieve from two candidate pools and then perform multi-hop reasoning to find answers.

to answer the question. For example, the question has low lexical overlap with the passage about the `'Lakers'`, and it needs the table as the bridge to retrieve this passage. Such cross-modality multi-hop retrieval features OTT-QA. More examples are displayed in Appendix.

OTT-QA is significantly different than the existing QA datasets. Existing table QA datasets (Pasupat & Liang, 2015; Yu et al., 2018; Chen et al., 2020) operates in the closed setting, so no retrieval is required, whereas most existing open QA datasets (Joshi et al., 2017; Yang et al., 2018) require only text retrieval, not table retrieval. One dataset, Natural Questions (Kwiatkowski et al., 2019) includes minor tabular information in its corpus, but the tables are nearly always of a restricted type (infobox tables). In contrast, OTT-QA models require retrieving *both* tabular data and text, and unlike the Natural Questions dataset, requires information fusion from text and tables in non-trivial ways.

OTT-QA poses great challenges to both the retriever and reader in open QA. Retrievers for OTT-QA need to consider two information forms, making the search space larger. Even worse, as questions in OTT-QA often require multi-hop inference, one round of retrieval is often not enough. Readers for OTT-QA also need to aggregate significantly amount of knowledge-intensive information, compared to other reader models: a single table in OTT-QA has an average length over 300 words. Moreover, readers are often expected to process multiple retrieved units due to uncertainty in retrieval, which makes it difficult to design strong reader models with limited length budget.

The baseline system that we propose to address these challenges uses an iterative retriever (Sun et al., 2019; Min et al., 2019; Ding et al., 2019), and a BERT reader (Devlin et al., 2019). The iterative retriever performs multiple rounds of auto-regressive retrieval over the table and passage corpus. Beam search is used to find multiple subsets of documents that may contain all the evidence required for an answer, and each subset is then fed to the BERT reader to predict the answer span. The highest-scored prediction is chosen as the answer. Unfortunately, this iterative retriever is computationally expensive, as the query representations need to be re-encoded at every retrieval step, and the reader is not always suitable for aggregating all the evidence.

We propose a more sophisticated system that addresses these challenges with two novel strategies: *fusion* retrieval and *cross-block* reading. The **fusion retriever** first pre-aligns a table segment to related passages, using a process similar to entity linking. Then, the pre-aligned table segments and retrieved passages are encoded together as a *fused block*, which contains information from both the table segment and related passages; hence, compared to the previous documents, it contains more richly contextualized information. We view the fused block as the basic unit to be retrieved, and instead of performing multiple runs of retrieval iteratively, the fusion retriever is used once to retrieve the top K fused blocks; however, due to errors in fusion, it may be that no single block contains all the necessary evidence. We thus also use a **cross-block reader** based on sparse-attention based transformer architecture (Ainslie et al., 2020; Zaheer et al., 2020), which can process long sequences. We use the cross-block reader to read all the top-K retrieved blocks, jointly, as a long sequence. Both strategies have proven effective compared to the baseline system: the best model using these strategies improves the exact match score of the baseline system from 10% to over 25%.

2 Background

The aim of an open QA system is to extract an answer to a question q from a given large corpus. Most open QA models are retriever-reader models, which extract answers in two steps: retrieval and reading. In the *retrieval* step, a retrieval model f is used to retrieve a set of passages from the text corpus. In the *reading* step, the reader model, is then used to extract answer from them.

Retrieval Function There are two commonly-used types of retrieval function f: sparse retrievers and dense retrievers. Our sparse retriever uses a unigram-based BM-25 score to retrieve an evidence block b from the candidate pool $\mathbb{B}$. Our dense retrieval function is a dual-encoder model (Bromley et al., 1994), and we follow (Lee et al., 2019; Guu et al., 2020) for the dual encoder design. The query and the passage are both encoded using a Transformer, which produces a vector for every token. As in (Devlin et al., 2019), the vector corresponding to the first token, `[CLS]` is used as a "pooled" representation of the sequence (denoted $\texttt{BERT}_{\texttt{CLS}}$). The dense retrieval function is the dot product between $\texttt{BERT}_{\texttt{CLS}}(q)$ and $\texttt{BERT}_{\texttt{CLS}}(b)$ for each evidence b in the candidate corpus—i.e., the scoring function is $f(q, b) = \texttt{BERT}_{\texttt{CLS}}(q)^T \texttt{BERT}_{\texttt{CLS}}(b)$, which can viewed as finding the nearest neighbor in vector space. In the multi-hop open QA setting (Yang et al., 2018), an iterative retrieval function (Sun et al., 2019; Min et al., 2019; Ding et al., 2019) is proposed, which defines the retrieval process as an auto-regressive formula. Our iterative retriever function is denoted as $f([q, b_1, \cdots, b_{j-1}], b_j)$, which appends the previous $j - 1$ rounds of retrieval to the original q in j-th round of retrieval. Similarly to autoregressive language models, one can use beam search of breadth-first search to explore the space of retrievals more broadly.

Single-Block Reader Due to the uncertainty in retrieval, the top-1 document might not always contain the answer. Existing models normally retrieve the top-k documents and feed them to the reader for span selection. The standard reader (Chen et al., 2017; Joshi et al., 2017) aims to extract a span from each of the retrieved blocks b_i and assign a confidence $p(b_i)p(a|b_i)$ to it, with $p(b_i)$ indicating the retrieval probability and $p(a|b_i)$ denoting the span selection probability. Multiple answers $\{a_1, \cdots, a_k\}$ are ranked with this confidence score and the highest scored answer span $\hat{a}$ is the final answer. Note that the reader needs to run k times, once for each of the top-k retrievals. We refer to this model as the *single-block reader* and use it as our baseline reader.

HybridQA HybridQA (Chen et al., 2020), a closed-domain QA dataset, is the most related to us. During the annotation of HybridQA, a table T and its relevant passages $\{P_1, \cdots, P_N\}$ (surrounding text and hyperlinked passage) are given to a crowd worker to write questions which necessarily require both the passage and table information to answer. The original dataset contains 72K multi-hop questions paired with 13K tables with their paired passages. During training/testing time, the ground-truth tables and passages are given to an HYBRIDER model to perform hops between table cells and passages to find the final answer. The HYBRIDER also serves as an important baseline in our paper. Since we do not have the 'ground-truth' table with its hyperlinks in an open-domain setting, we must reconstruct the input table with a retrieval-based method, as outlined below.

3 Task and Dataset

Problem Definition In OTT-QA, the retrieval corpus consists of a set of table candidates $\mathbb{B}_T$ and a set of passage candidates $\mathbb{B}_P$. The task is to answer question q by extracting answer strings from blocks $b \in \mathbb{B}_T \cup \mathbb{B}_P$, where b can be either textual and tabular data. We adopt the standard exact match (EM) and F1 scores (Yang et al., 2018) for evaluation. Different from HybridQA, OTT-QA's table candidates are more general web tables without given hyperlinks. This decision was made to make the problem setting more general, as otherwise systems that solve OTT-QA could only be applied to high-quality hyperlinked tables such as Wikipedia tables. However, in OTT-QA, we do allow the use of these hyperlinks as additional training signals. Removing links in table candidates makes the overall task much more challenging, but making the final systems applicable to general tables as well. Thus, an OTT-QA model needs to jointly retrieve both tables and text, and then aggregate this heterogeneous evidence to find the answer span.

Candidate Pool For our table collection $\mathbb{B}_T$, we extracted all the regular tables with their metadata including page title, page section title, and section text. The meta data denoted T_M, is essential for de-contextualization. We obtain a table corpus containing over $400k$ high-quality tables with an average length of 320 words including metadata. For the text passage collection $\mathbb{B}_P$, we crawl English Wikipedia dump pages and filter out noisy pages, we follow HybridQA (Chen et al., 2020) and only keep a maximum of 12 sentences in the introduction section as the passage. We obtain a corpus containing over 5 million passages, with average 94 words.

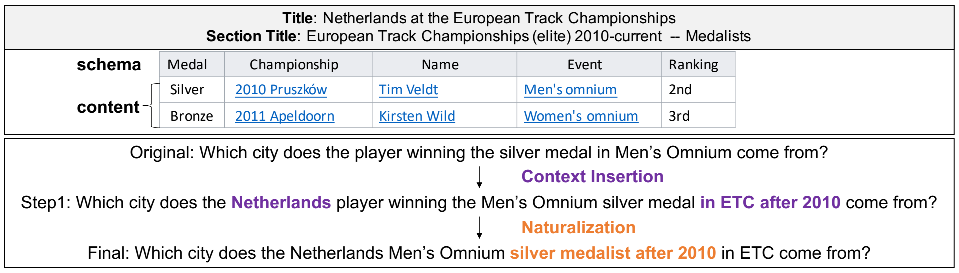
Table Decomposition In OTT-QA, a single complete table with structured representation (Herzig et al., 2020) can easily exceed the 512-token limit, which poses great challenges to the downstream reader to deal with top-K retrieval. Hence we propose to decompose each table T into multiple rows R_i as our basic retrieval block. Each row R_i block also retains the headers, metadata, and global max/min information from the original table, which is denoted **table segment**. This decomposition procedure brings the size of table candidate $\mathbb{B}_T$ from $400k$ to 5 million, making the retrieval problems even more fine-grained and more challenging. Our table segment representation is described in Appendix subsection A.5. In summary, we build a candidate pool of 5 million table segments $\mathbb{B}_T$ and a candidate pool of 5 million passages $\mathbb{B}_P$. We denote as $\mathbb{B}$ as our full candidate pool, which our model needs to find the block b (a table segment or a passage) containing the answer span.

3.1 Question and Answer Annotations

Our question and answer pairs are built upon the existing HybridQA (Chen et al., 2020) dataset, with several significant changes. First, crowd workers 'decontextualize' the questions so that they are not under-specified or context-dependent, and thus suitable for the open setting. Second, we add more questions to the development/test set to remove possible annotation bias. During annotation, we adopt strict quality control[2] and more details are described in Appendix section A.1.

Decontextualization Most questions in HybridQA are contextualized with a given table and several passages, with corresponding questions written by crowd workers. Often, the crowd-sourced questions assume the context. For example, the questions might contain the words `"the players"` because the given table is about `"Netherlands players"`. We thus needed 'de-contextualize' (Parikh et al., 2020) the original context-dependent questions, so they could serve as standalone questions, specific enough to imply a unique answer relative to the corpus.

To discourage excessive unwanted modification, we enforce a two-step annotation procedure, as depicted in Figure 2. In the first phase, the worker is only allowed to insert words or phrases (or replace pronouns) into the questions based on the information provided by Wikipedia Title, Section Title, and Section Text. After this step, we often potentially obtain overly-complicated questions that are artificial and unnatural in the real world. Therefore, We manually selected the worst 25% questions and sent these questions to annotators to make them more concise and natural.


Title: Netherlands at the European Track Championships
Section Title: European Track Championships (elite) 2010-current -- Medalists

schema	Medal	Championship	Name	Event	Ranking
content	Silver	2010 Pruszków	Tim Veldt	Men's omnium	2nd
	Bronze	2011 Apeldoorn	Kirsten Wild	Women's omnium	3rd

Original: Which city does the player winning the silver medal in Men's Omnium come from?
↓ **Context Insertion**
Step1: Which city does the **Netherlands** player winning the Men's Omnium silver medal **in ETC after 2010** come from?
↓ **Naturalization**
Final: Which city does the Netherlands Men's Omnium **silver medalist after 2010** in ETC come from?

Figure 2: The 'de-contextualization' annotation phase of OTT-QA. In the first step, the annotator is restricted to add phrases from the context. In the second step, the annotator is specifically requested to make the sentence more concise and natural.

Additional Evaluation Examples As all the questions from HybridQA are based on the $13k$ tables from the HybridQA set, so no questions are asked about the newly crawled $400k$ tables. This potentially generates unwanted statistical biases or artifacts for the model to exploit, and potentially bias the final evaluation results. Therefore, we randomly sampled another 1100 tables from the newly crawled tables, and follow the original annotation process used by HybridQA to re-collect 2200 new questions. These new questions were mainly used in the dev/test set to alleviate biases. Below we refer to the subset of tables used by original HybridQA as the *in-domain* tables.

[2]The collection is conducted on an established crowdsourcing platform with annotators from countries with English as the native language. The annotators were required to meet the requirements: 1) native speaker in an English-speaking country. 2) having an approval rate of over 95%. 3) having at least 500 approved jobs.

Distant Supervision Signals For the in-domain tables ($\approx 8k$), the cell-wise hyperlinks are provided in OTT-QA as a potential signal for supervision. We use $H_{i,j} = \{b_1, b_2, \dots \in \mathbb{B}_P\}$ to denote the hyperlinks in cell $T_{i,j}$. Since in HybridQA the oracle fine-grained answer span is not explicitly annotated, we approximate this by traverse the table and hyperlinked passages to find any exact matches. This process contains some noise—a manual study reveals that it roughly contains 15% error. We use this 'weakly-supervised' fine-grained information to train our models. We denote the 'approximate' block of the answer span for answer a as b_a and use it to train our model.

3.2 Dataset Statistics

After annotation, we sampled roughly 2K questions from the in-domain HybridQA dataset, and then mix them with the newly collected out-domain questions to construct our dev and test sets. Finally, we have 41,469 questions in the training set, 2214 questions in the dev set, and 2158 questions in the test set. We conduct more in-detailed analysis over the reasoning types and show them in the Appendix A.2, a remarkable difference from original HybridQA is that a proportion of questions actually have multiple plausible inference chain under open-domain setting.

4 Model

Our model for OTT-QA is a retriever-reader model with new designs for both retriever and reader. As discussed briefly above, we propose to use a fusion retriever instead of using a standard iterative retrieve, and we also propose to use cross-block readers to replace a standard single-block reader.

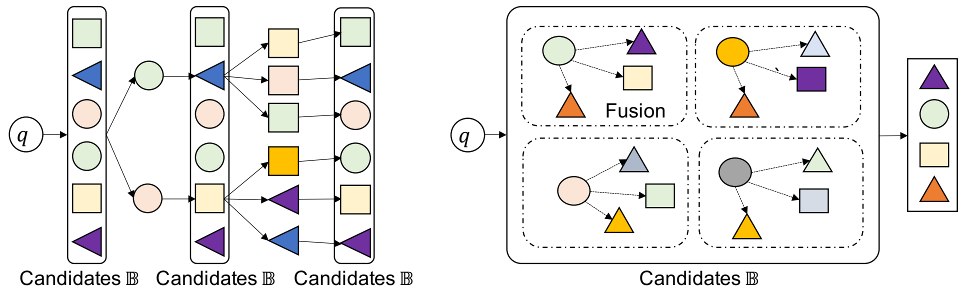


Figure 3: **Left:** Iterative retrieval (baseline). **Right:** Fusion retrieval.

4.1 Fusion Retriever

Iterative retrieval (Figure 3, Left) has the following issues. First, iterative retrieval training often requires having supervision signals for every retrieval step to reach good performance, which is not available in OTT-QA. The iterative retrieval also suffers from the problem of error propagation, as early mistakes can propagate to later retrieval stages. Finally, the computation cost for applying a dual-encoder for iterative retrieval is very high, for every stage, the query embedding has to be re-encoded to include the entire retrieval history.

We propose an alternative strategy to replace multi-step retrieval, namely fusion retrieval (Figure 3, Right). In fusion retriever, we first use an 'early fusion' strategy to group relevant heterogeneous data before retrieval. The fusion procedure augments each block with more context from related blocks with different modalities, which provides more clues for the retriever to utilize. Early fusion is very important for retrieving table segments, which often have little context by themselves.

The early fusion process aims to fuse a table segment and relevant passages into a group. Here we propose to fuse entities mentioned in a table segment to the appropriate passages for those entities: this is similar to document expansion based on a traditional entity linking step. The problem is challenging due to the mismatch between the lexical forms from the table (which for brevity are often abbreviated) and the relevant passage titles. For example, a cell in the table of `"NCAA Division I Men's Football Tournament"` contains the term `"Penn State"`. Directly matching `"Penn State"` against the passage corpus will lead to `"Penn State University"` rather than the ground-truth hyperlinked entity, named `"Penn State Nittany Lions football"`. Therefore,

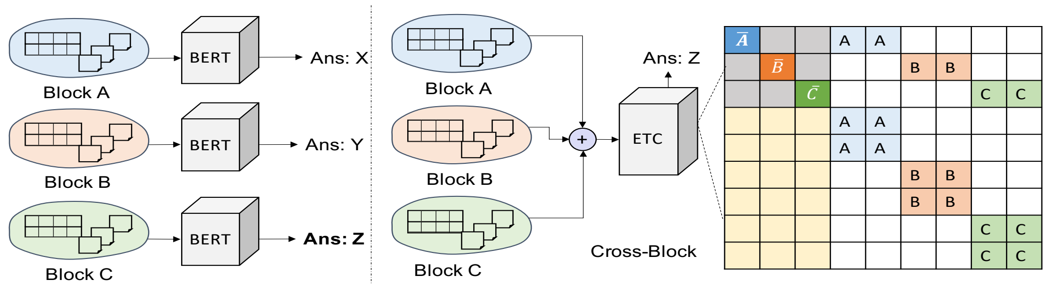


Figure 4: **Left:** Single-block reader (baseline). **Right:** Cross-block reader, with $\bar{A}$ denoting the global state corresponding to local block A.

we propose an additional query augmentation step, which takes in a table segment block b_T and generates a sequence of augmented queries $q_1, q_2, \cdots, q_n$ token by token to make the queries more similar to the passage title. The augmented queries are then used to search for nearest neighbors in the passage corpus $\mathbb{B}_P$ using BM25 as the final entity linking step, which is depicted in Appendix subsection A.4. The query augmentation is implemented with a GPT-2 model (Radford et al., 2019), fine-tuned on the supervised pairs of (table segment, hyperlink) from the in-domain tables. Each b_T is fed to find its companions $b_P^1, \cdots, b_P^n$, they are collectively called b_F.

The document embedding of the fused block is produced by cross-attention between the table and the text units with a BERT encoder: $\texttt{BERT}_{\texttt{CLS}}(b_F) = \texttt{BERT}_{\texttt{CLS}}([b_T, b_P^1, \cdots, b_P^n])$. The fused embedding contains richer context from both modalities, tables, and text. Moreover, we can save computation by computing the representation of the fused blocks offline prior to training the reader.

To enhance the neural retrieval system to retrieve fused blocks, we also apply the Inverse Cloze Task (ICT) (Lee et al., 2019) pretraining task on the corpus of fused blocks. ICT is a way to generate pseudo-training data for dense retrieval. Unlike standard document-wise ICT, our fused block contains both table segments and multiple passages. Given a fused block b_F, we generate the pseudo-query in the following way: we first corrupt the table segment by randomly dropping half of the words from the table metadata and cells to obtain a corrupted table segment $\hat{b}_T$. We then randomly sample a sentence $\hat{b}_P$ from the companion passage. We combine $\hat{b}_T$ and $\hat{b}_P$ as a pseudo query, which is trained to retrieve the original fused block b_F. We pre-train the dual encoder with the generated ICT-style pseudo data to enhance its ability in matching lexically similar documents. After pre-training, the retriever is fine-tuned on OTT-QA. Finally, at inference time, the retriever is used to retrieve the top K fused blocks for a question, which are then passed to the reader.

4.2 CROSS-BLOCK READER

The reader typically needs to process the top-k retrieved blocks returned by the retriever in order to extract the best answer, as the top-1 block might not contain enough evidence to answer the question. As demonstrated in Figure 4, cross-block reader aims to address this issue by using cross attention between different blocks to model their dependencies. To obtain the cross-block reader, we take the pre-trained long-range sparse attention transformer (ETC) (Ainslie et al., 2020), which can accept up to 4096 tokens as input, and then fine-tune the model on the distant supervision data. During training, the ground truth (fused) blocks are mixed with the hard negative blocks from the retriever. We take the top-k retrieval results to fill the 4096 token space (roughly 15 fused blocks).

Cross-attention between blocks allows a much more powerful way to aggregate information across the k retrieved blocks compared to single-block reader, especially when the blocks are fused. This is feasible because the design of the attention structure in ETC, which can constrain the attention of each token to its neighboring tokens within a local radius in its local block. Such sparse attention can decrease the attention computation complexity from quadratic $\mathcal{O}(N^2)$ to linear $\mathcal{O}(N|R|)$, where $|R|$ is the local radius (where $N = 4096$ and $|R| = 84$ in our experiments). To allow cross-block interaction, ETC assigns a global state for each local block in the long sequence, and blocks can attention to each other through multiple layers of such global-local structures.

Retriever	Dev-Sparse		Dev-Dense		Test-Best	
Model	EM	F1	EM	F1	EM	F1
HYBRIDER (Chen et al., 2020)	7.2	10.4	7.4	10.2	7.6	10.8
Iterative-Retrieval + Single-Block Reader	9.8	13.3	7.9	11.1	9.6	13.1
Fusion-Retrieval + Single-Block Reader	14.3	17.8	13.8	17.2	13.4	16.9
Iterative-Retrieval + Cross-Block Reader	19.0	23.6	14.4	18.5	19.4	24.0
Fusion-Retrieval + Cross-Block Reader	24.6	28.1	**25.8**	**30.2**	**26.0**	**30.5**

Table 1: **Main Results**. We conduct experiments with both sparse and dense retrievers using the dev set, and then select the best setting to report the test set results (as indicated by the word "Best"). Fusion-Retriever and Cross-Block Reader both brings significant improvement.

5 EXPERIMENTS

All of our code is based on Tensorflow (Abadi et al., 2016). For the retriever part, the sparse retriever is built on top of DrQA[3] with unigram features, and the dense retriever is built on top of ORQA[4]. The single-block retriever is based on BERT-uncased, and the cross-block reader is based on ETC (Ainslie et al., 2020). Both of them consist of 12 layers with a hidden size of 768, the minor differences in the relative positional embedding used in ETC. All the models are trained with a learning rate of 1e-5 optimized by AdamW (Loshchilov & Hutter, 2019). We use in-batch negatives (Lee et al., 2019) to train our dense retrievers. A more detailed implementation of the baseline iterative retriever is described in Appendix subsection A.6.

In fusion retriever, we use the 'fused' block containing the 'approximate' answer block b_a as the positive instance. In iterative retriever, since the auto-regressive model $p(b_j|q, b_1, \cdots, b_{j-1})$ requires fine-grained inference chain for step-wise supervision, which is not given in OTT-QA. We apply lexical match based heuristics to synthesize inference chains as weakly supervised training data (described in Appendix). For both all the dense retrievers, we pre-train with 10K steps using the generated pseudo query and then fine-tune them another 10K step using a batch size of 2048. For the cross-block reader, we fine-tune with a batch size of 64. Both are using 16 cloud TPUs.

Results The main results are presented in Table 1. We consider HYBRIDER (Chen et al., 2020) from closed domain HybridQA as our first baseline. Since this model requires a ground truth table with its hyperlinks to do modularized reasoning, we use BM25 to retrieve the most relevant table and passages to reconstruct an 'approximated' table as the input for the HYBRIDER.

The results from different designs of retriever and readers investigated in this paper are as follows. First, we follow the standard framework by combining iterative retriever with the standard single-block reader. This baseline obtains 9.8% EM, a decent improvement over HYBRIDER. By replacing the iterative retriever with the proposed fusion retriever, the EM score can be achieve 14%, which demonstrates the effectiveness of the early fusion strategy. By further replacing the single-block with the proposed cross-block reader, the score can be further boosted to over 25% for both sparse and dense setting. These improvements demonstrate the effectiveness of our novel strategies.

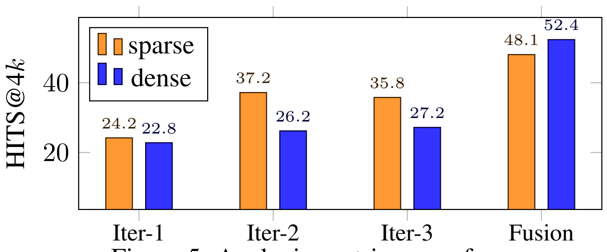


Figure 5: Analyzing retriever performance.

To understand the results more, we compare the performance of the iterative and fusion retrievers in Figure 5. HITS@4K is used to measure the retriever performance, which indicates the chance of ground truth answer appearing in the retrieved 4096 tokens, which reflects an upper limit QA system using our reader. We also experiment with sparse and dense iterative retrievers with different numbers of steps to show the necessity of multi-hop reasoning in OTT-QA.

[3]https://github.com/facebookresearch/DrQA
[4]https://github.com/google-research/language/tree/master/language/orqa

We observe that the 1-step retrieval (Iter-1) has a unanimously lower recall for both sparse and dense settings due to the multi-hop nature of the OTT-QA. Adding the second retrieval hop (Iter-2) can greatly improve the recall, but adding the third retrieval hop has very little impact. In contrast to the iterative retriever, the fusion retriever only retrieves once over the pool. It greatly improves the performance over the iterative setting, the sparse setting can rise from 35.8% to 48.1%, which indicates the grouped blocks contain more lexical match than standalone blocks. The dense retriever's improvement is even more dramatic (from 27.2% to 52.4%). Unlike the iterative retriever, which trains on the heavily noisy 'synthesized' inference chain data, the fusion retriever only requires the answer block as the supervision signal, which makes the model less prone to noise. To better understand the retriever, we conduct error analysis for the retriever and conclude most popular types: 1) low lexical overlap, 2) fusion errors, 3) numerical reasoning, 4) distraction terms. More detailed examples are shown in subsection A.8.

We also observe that the single block reader cannot handle the multiple retrieval units well. When we increase the retrieved tokens from 512 $\rightarrow$ 1024 $\rightarrow$ 4096 and chunk them to multiple blocks for the reader, the final EM score actually drops from 13.8 $\rightarrow$ 10.7 $\rightarrow$ 7.8. We conjecture that the local soft-max normalization inside the local block makes the reader overly confident about its prediction and cause exposure bias issue in the prediction.

6 Related Work

KB and Text: The problem combining structured and unstructured data has been studied in question answering. The previous approaches are mainly focused on leveraging knowledge graph triples to assist the textual question answering like FusionNet, PullNet (Sun et al., 2018; 2019), and Knowledge-Guided Retrieval (Min et al., 2019). These approaches use existing KB to guide textual document retrieval from the web. However, KB is mainly used as an alternative tool, rather than a necessary information source in the experimented benchmarks. Our generative entity linker is related to knowledge-enhanced language understanding (Petroni et al., 2020), which proposes seq2seq model to deal with different knowledge-intensive tasks like slot filling, entity linking, etc.

Table Understanding: Tables have been a ubiquitous information representation form to express semi-structured information. There has been a long-standing effort to utilize tables in natural language processing applications (Pasupat & Liang, 2015; Zhong et al., 2017; Yu et al., 2018; Parikh et al., 2020; Chen et al., 2019). However, these existing tasks are restricted to in-domain cases without requiring any retrieval, our paper is the first to investigate retrieving web table for downstream tasks. Another line of related works are TAPAS (Herzig et al., 2020) and TABERT (Yin et al., 2020), which investigates joint pre-training over textual and tabular data. Our method draws inspiration from these models to use special tokens and embeddings to encode spatial and logical operations inside tables.

Long Range Transformer: Recently, many transformer variants to resolve the $\mathcal{O}^2$ attention cost have been proposed including Sparse Attention (Child et al., 2019), Reformer (Kitaev et al., 2020), Routing Transformer (Roy et al., 2020), Longformer (Beltagy et al., 2020) and ETC (Ainslie et al., 2020). These different transformer models apply hierarchical architecture, local-sensitive hashing, global-local state to decrease the attention complexity to nearly linear. Our cross-block reader is based on ETC (Ainslie et al., 2020), unlike ETC to process one long document for question answering, our task requires reading multiple blocks containing both structured and unstructured data.

7 Conclusion

We focus on the problem of performing open question answering over tables and text in this paper. One interesting question we would like to ask in the future is: can we extend open question answering system to more modalities? Some questions can be better answered by images and other resources, but the task can be drastically more challenging by including more modalities, as we have learned from this paper. Finally, we believe the techniques we proposed might be useful for other open-QA setting, especially the comparisons between iterative retriever and fusion retriever.

REFERENCES

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pp. 265–283, 2016.

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Philip Pham, Anirudh Ravula, and Sumit Sanghai. Etc: Encoding long and structured data in transformers. *Proceedings of EMNLP 2020*, 2020.

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

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, pp. 737–744, 1994.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1870–1879, 2017.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. Tabfact: A large-scale dataset for table-based fact verification. In *International Conference on Learning Representations*, 2019.

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

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2019.

Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. Cognitive graph for multi-hop reading comprehension at scale. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2694–2703, 2019.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Realm: Retrieval-augmented language model pre-training. *Proceedings of ICML 2020*, 2020.

Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. Tapas: Weakly supervised table parsing via pre-training. *ACL 2020*, 2020.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, 2017.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. *EMNLP 2020*, 2020.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *ICLR*, 2020.

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

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

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *ICLR 2019*, 2019.

Sewon Min, Danqi Chen, Luke Zettlemoyer, and Hannaneh Hajishirzi. Knowledge guided text retrieval and reading for open domain question answering. *arXiv preprint arXiv:1911.03868*, 2019.

Ankur P Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. Totto: A controlled table-to-text generation dataset. *arXiv preprint arXiv:2004.14373*, 2020.

Panupong Pasupat and Percy Liang. 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)*, pp. 1470–1480, 2015.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vassilis Plachouras, Tim Rocktäschel, et al. Kilt: a benchmark for knowledge intensive language tasks. *arXiv preprint arXiv:2009.02252*, 2020.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

Stephen Robertson and Hugo Zaragoza. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc, 2009.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *arXiv preprint arXiv:2003.05997*, 2020.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4231–4242, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1455. URL `https://www.aclweb.org/anthology/D18-1455`.

Haitian Sun, Tania Bedrax-Weiss, and William Cohen. PullNet: Open domain question answering with iterative retrieval on knowledge bases and text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2380–2390, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1242. URL `https://www.aclweb.org/anthology/D19-1242`.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, 2018.

Wen-tau Yih, Kristina Toutanova, John C Platt, and Christopher Meek. Learning discriminative projections for text similarity measures. In *Proceedings of the fifteenth conference on computational natural language learning*, pp. 247–256, 2011.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. Tabert: Pretraining for joint understanding of textual and tabular data. *ACL 2020*, 2020.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 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*, pp. 3911–3921, 2018.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *arXiv preprint arXiv:2007.14062*, 2020.

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

A Appendix

A.1 Dataset Annotation

Filtering The original HybridQA dataset contains over $72k$ questions paired with $13k$ hyperlinked tables. We adopt two filtering heuristics to make the decontextualization easier. First, we filter out tables without enough meta-information, or containing too much non-textual information[5]. Secone, we filter out overly-long questions, i.e., questions longer than 30 words. These two filtering heuristics result in a cleaner subset of $46k$ questions paired with $9k$ in-domain tables.

Quality Control During annotation, we conduct strict manual quality evaluation over the decontextualized, with the following criteria: 1) the annotated question retains the same semantics and answer as before, 2) the annotated question still requires multi-hop reasoning over both table and passages. 3) the annotated question is concise and fluent. The manual quality checking was performed over batches distributed to the same annotator. Each batch consists of six questions, one of which will be sampled to decide the acceptance/rejection of this whole batch. The overall acceptance rate for the crowd-sourcing job is 71%, and a rejected job was re-distributed until it was accepted.

A.2 Dataset Analysis

We randomly sampled 100 questions from the dataset to manually analyze the kinds of inference chains seen in OTT-QA and divide the major types into the following categories:

1. Single hop questions (13%) require reading one table or one passage to answer.
2. Two hop questions (57%) require reading one passage and one table to answer. These can be subclassified as 'table bridge' $\rightarrow$ 'answer text'[6] or 'text bridge' $\rightarrow$ 'answer table'.
3. Multi hop questions (30%) require reading two passages and one table to answer. These mainly following the reasoning chain of 'text bridge' $\rightarrow$ 'table bridge' $\rightarrow$ 'answer text'.
4. Questions with multiple reasoning paths: Due to information redundancy in Wikipedia, similar information can appear in both table and text. We find that 9% of questions are answerable by reading one text passage, 18% of questions are answerable by reading two text passages and 4% of questions are answerable by reading two tables.

A.3 Dataset Examples

We demonstrate more examples in Figure 6, which includes more diverse inference chain. Like table $\rightarrow$ text; text $\rightarrow$ table $\rightarrow$ text; text + text $\rightarrow$ comparative $\rightarrow$ table. Our model is able to perform these reasoning types quite well by jointly matching query against the fused table-text block.

A.4 Fusion Retriever

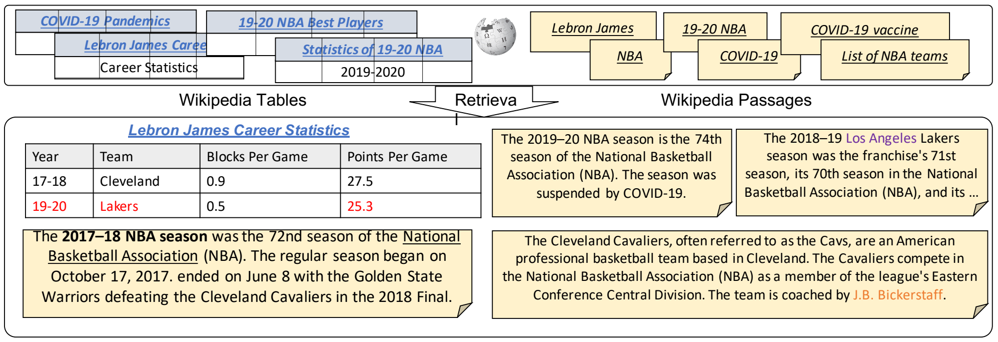
The entity linker procedure from table segment to passages are described as Figure 7.

A.5 Table Segment Representation

The table decomposition is visualized as Figure 8. The title/section title are prefixed in the table segment. We add row position token '1st', max/min special token over the column to infuse the global table information back to the segmented unit. The column embedding is added as another vector to the representation. The table segment representation is relatively small and easy to deal with in the following reader model.

[5]like longitude, latitude, mathematical formula, etc

[6]I.e., a table forms a bridge between a question and a textual passage, which is read in the first hop, and the answer is extracted from the text.



Year	Team	Blocks Per Game	Points Per Game
17-18	Cleveland	0.9	27.5
19-20	Lakers	0.5	25.3

The **2017–18 NBA season** was the 72nd season of the National Basketball Association (NBA). The regular season began on October 17, 2017. ended on June 8 with the Golden State Warriors defeating the Cleveland Cavaliers in the 2018 Final.

The 2019–20 NBA season is the 74th season of the National Basketball Association (NBA). The season was suspended by COVID-19.

The 2018–19 Los Angeles Lakers season was the franchise's 71st season, its 70th season in the National Basketball Association (NBA), and its ...

The Cleveland Cavaliers, often referred to as the Cavs, are an American professional basketball team based in Cleveland. The Cavaliers compete in the National Basketball Association (NBA) as a member of the league's Eastern Conference Central Division. The team is coached by J.B. Bickerstaff.

Q1: How many points per game did Lebron James get in the COVID-19 NBA Season?	A1: COVID-19 -> 19-20 Season -> 25.3
Q2: Who is the coach of the team that Lebron James played in to achieve his highest score in his career?	A2: 27.5 -> Cleveland -> J. B. Bickerstaff
Q3:What suspends the NBA season during which Lebron James has an average points per game of 25.3?	A3: 25.3 -> 19-20 Season -> COVID-19
Q4:For the season suspended by COVID-19 and the season defeated by Warriors in Final, which season has Lebron obtained more points?	A4: 25.3 < 27.5 -> 17-18

Figure 6: More examples from OTT-QA

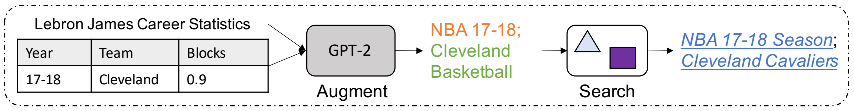
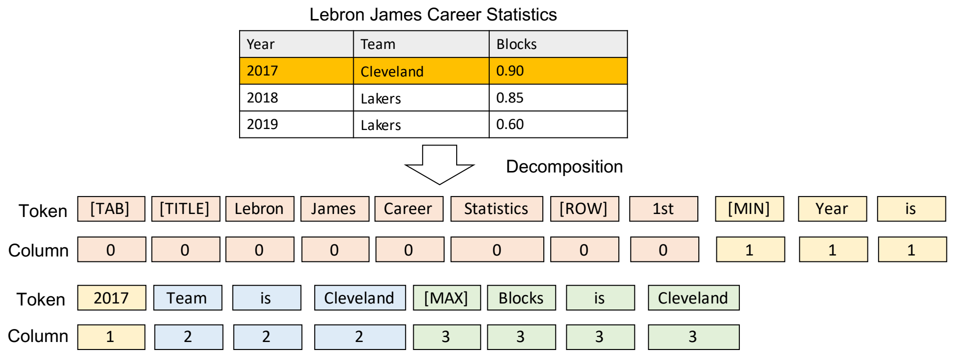


Figure 7: Fusion: 1) GPT-2 query augmentation, 2) nearest neighbor search over passages.



Year	Team	Blocks
2017	Cleveland	0.90
2018	Lakers	0.85
2019	Lakers	0.60

Token	[TAB]	[TITLE]	Lebron	James	Career	Statistics	[ROW]	1st	[MIN]	Year	is
Column	0	0	0	0	0	0	0	0	1	1	1
Token	2017	Team	is	Cleveland	[MAX]	Blocks	is	Cleveland			
Column	1	2	2	2	3	3	3	3			

Figure 8: The decomposition of the original table into segments.

A.6 Iterative Retriever

Iterative retrieval has been used in recent graph-based multi-hop retrieval models to gradually retrieve documents to find the correct supporting evidence. Specifically, the retriever condition the i-th round retrieval on the previous round of retrieval results.

Sparse Retriever The sparse retriever uses un-gram lexical feature to compute the BM-25 score between the $q, .., b_{1..j-1}$ over $b_j \in \mathbb{B}$ to get the topk candidates. Here we describe a two step LxM retrieval procedure, in the first step, the model calculates BM25 score between question over all the candidates in $\mathbb{B}$ to select top L/2 table segments b_T, and L/2 passages b_P. In the second step, the question is concatenated with the retrieved table segment to form L/2 new queries $[q; b_T]$ to retrieve LM/2 passages from $\mathbb{B}$. The question is also concatenated with the retrieved passage titles to form another K/2 queries $q; b_P$ to retrieve LM/2 table segments. The LxM retrieval procedure results in at most LM+L unique blocks, each unique block aggregates its score from two rounds denoted as $p(b)$, which is used to rank the topK candidates for the next step.

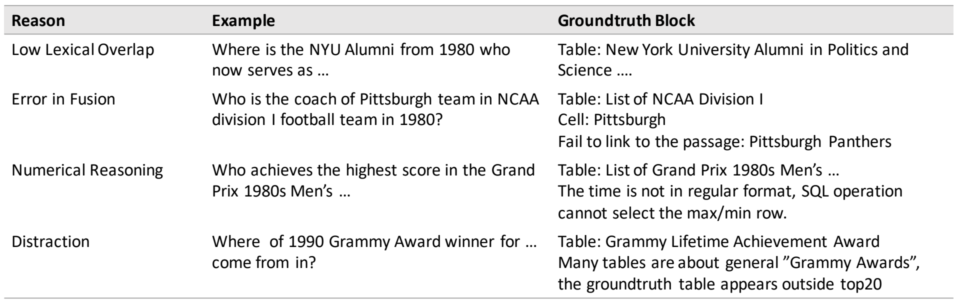
Dual-Encoder Retriever The dual encoder uses BERT-based encoder to compress each question, table segment and passage into a fixed length vector and then compute the dot product between fixed vectors to obtain the highest scored candidates from pool $\mathbb{B}$. However, since the dataset does not provide explicit supervision signal for the iterative retrieval, we heuristically synthesize some noisy retrieval chain using lexical matching (discuss in Appendix). The retrieval inference chain is depicted as $b_1 \rightarrow b_2 \rightarrow b_K$, which is used to train the model $p(b_k|q, b_{1..k-1})$ in a supervised manner. At inference time, the dual encoder retriever will encode a query q into fixed vector and retrieve first L blocks from $\mathbb{B}$. The blocks are appended to query q to form L new queries $[q; b_i]$, which is re-encoded and search for LM new neighbors. We experiment with maximum of 3-step retrieval of LxMxN to obtain maximum of L+LxM+LMN unique blocks. Similarly, each unique block aggregates its score from different rounds to select the topK candidates for the next step.

A.7 DENSE RETRIEVAL/IN-BATCH NEGATIVE

Recently, different dense-retrieval (Lee et al., 2019; Guu et al., 2020; Karpukhin et al., 2020) based on dual-encoder Bromley et al. (1994) have been shown to surpass traditional sparse retrieval in open-QA models. The query and the passage are both encoded using Transformer, which produces a vector for every token. As in (Devlin et al., 2019), the vector corresponding to the first token, `[CLS]`, which is used as a "pooled" representation of the sequence (denoted $\texttt{BERT}_{\texttt{CLS}}$). The dense retrieval function can be represented as the dot product between $\texttt{BERT}_{\texttt{CLS}}(q)$ and $\texttt{BERT}_{\texttt{CLS}}(p)$ for each document in the text collection. like TF-IDF (Chen et al., 2017) and BM25 (Robertson & Zaragoza, 2009) on some Open QA datasets. To train the dual-encoder, the in-batch negative trick (Yih et al., 2011; Karpukhin et al., 2020) plays an important role, which uses B training instances in each batch and view the rest B-1 instances inside the batch as the negative. In this way, the model reuses computation and effectively train on B^2 question/document pairs in each batch.

A.8 ERROR ANALYSIS

We conduct error analysis to see what are the main issues with the retriever and conclude the following major types in Figure 9. The major issues are low lexical overlap, fusion errors, numerical reasoning and distraction words to retrieve unrelated evidences.



Reason	Example	Groundtruth Block
Low Lexical Overlap	Where is the NYU Alumni from 1980 who now serves as …	Table: New York University Alumni in Politics and Science ….
Error in Fusion	Who is the coach of Pittsburgh team in NCAA division I football team in 1980?	Table: List of NCAA Division I Cell: Pittsburgh Fail to link to the passage: Pittsburgh Panthers
Numerical Reasoning	Who achieves the highest score in the Grand Prix 1980s Men's …	Table: List of Grand Prix 1980s Men's … The time is not in regular format, SQL operation cannot select the max/min row.
Distraction	Where of 1990 Grammy Award winner for … come from in?	Table: Grammy Lifetime Achievement Award Many tables are about general "Grammy Awards", the groundtruth table appears outside top20

Figure 9: The main error types in the retriever.